A LOOP NETWORK USING INTEL'S 8274-MPSC

A Thesis Submitted
In Partial Fulfilment of the Requirements
for the Degree of

MASTER OF TECHNOLOGY

by

SIBY ABRAHAM

to the

DEPARTMENT OF ELECTRICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY, KANPUR

DECEMBER, 1986

EE-1986-M-ABR-LOO

dedicated to

my beloved parents

CERTIFICATE

Certified that the work entitled 'A loop network using Intel's 8274-MPSC' has been carried out under my supervision and has not been submitted elsewhere for a degree.

(Sanjay K. Bose)
Assistant Professor
Department of Electrical Engg.
I.I.T. Kanpur.

ACKNOWLEDGEMENTS

It is with immense gratitude and deep sense of indebtedness that I acknowledge the valuable suggestions, help and guidance of Dr. S.K. Bose. But for his active interest, enthusiastic guidance and whole hearted co-operation, this effort would not have materialised.

I take this opportunity to express my sincere and heart-felt gratitude to Mr. G.N.M. Sudhakar, for all his interest, help and co-operation, during the course of this project. A name, I can remember only with a profound sense of indebtedness is that of Balamuraleedhar, my colleague, who has been there always, ready and happy to help in all the ways he can. My thanks are also due to Messrs Sajan Jacob, Prem Malhotra, B. Nandi and Bhatnagar for their co-operation.

Finally I express my appreciation and thanks to Mr. C.M. Abraham for his efficient and meticulous typing.

ABSTRACT

A SDLC loop network has been designed and implemented using Intel's 8274-MPSC. Transmissions and receptions in the loop are under DMA control. Each node gets its chance to transmit, when it captures the circulating token. Host processors and terminals are hooked on to the basic loop, through RS-232 interfaces at each of the IMP nodes. Two such stations have been made and tested. A data rate of 192 KBs for the loop and 2.4 KBs for the IMP-Host interface have been achieved.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Earliest attempts at computer communication were single processor batch systems that supported a few local terminals and 'channel couplers' [1] in the 1960's. IBM/360 running BTAM, SNA generation 1 and 2 releases (with programmable communication functions) and ARPANET came later [1]. Today we have 'inter-connected collections of autonomous computers' [2] capable of peer-to-peer exchanges of information.

1.1 CHARACTERISATION OF A NETWORK

The basic function of a computer network is to provide 'access paths' between the end users [3]. By access paths, we mean, the sequence of functions that makes it possible for one end user to be not only physically connected to another, but to actually communicate. This should take care of errors of various types and large differences in speed, format, patterns of intermittency etc. that are characteristic to each end user.

1.2 LOCAL AREA NETWORKS

The distinct features that characterize a LAN are :

(i) geographical span of no more than a few km.

(ii) comparatively high data rates

(iii) ownership of a single organisation
and (iv) absence of any legal bindings on the transmission medium.

Load and resource sharing capabilities and amenability to remote data acquisition and processing are distinct advantages of LANS. Better reliability of the overall system and the flexibility to add more processors depending on demand, make LANs preferred over highly localized main frames. Expectation of a simpler software design, in distributed computing is yet another feature [2].

Because of the aforementioned advantages and the superior price/performance ratio of small computers over large ones, LANs have come to stay in office and industrial environments. Sharp decline in the price of minicomputers as compared to communication devices have speeded up this process. IEEE has adopted and standardised Ethernet CSMA/CD and token ring as two possible alternatives for high speed LANs. However, other configurations like star, tree, loop etc. also exist.

1.3 ORGANISATION OF SUBSEQUENT CHAPTERS

The second chapter broadly outlines the various design considerations involved in the development of the proposed LAN. The actual hardware design aspects, with circuits and timing diagrams, are highlighted in the third chapter. The fourth chapter covers the salient features of the application software. The concluding chapter discusses the performance details and suggests some enhancements.

CHAPTER II

DESIGN CONSIDERATIONS

The attempt here is to design and develop a LAN, interconnecting 8085-based workstations, striking a viable compromise between the data rates possible and expenses involved. It is intended that this basic LAN should serve as the backbone structure (IMP-to-IMP protocol) for interconnecting different host processors with information exchange capability. The various design considerations that influenced the development are given below .

2.1 OBJECTIVES AND OPTIONS

2.1.1 Selection of DLC Protocol

The choice here is between Asynchronous or synchronous character oriented protocols like Bisync and Synchronous bit-oriented protocols like SDLC. Asynchronous type protocols are the simplest to implement but have the disadvantages of being rather inflexible and inefficient. Line capacity is wasted in adding to each character various fixed bit patterns for synchronisation. Synchronous character oriented protocols, though better than Asynchronous, still retain much of such disadvantages. These use same alphabet set and same positions in a frame for line/device control characters and text characters. Every time

a new choice of the alphabet (e.g.,5 bits/character, 8 bits/character etc.) is made for the peculiar needs of one end user, a new and different variant of line control is needed [3].

Bit oriented protocols alleviate the above difficulties as well as problems of bit efficiency. In these protocols, line control information always occurs at its own place in a frame. The line control commands are specific bit patterns that have nothing to do with the alphabet set. The inherent advantages of these protocols are :

(i) code independence
(ii) adaptability to various configurations and applications
(iii) full duplex capability
(iv) high efficiency
and (v) high reliability [4].

For the present implementation the popular SDLC protocol have been chosen, since it is fairly simple to implement. Various LSI controllers are also available which support this protocol easily.

The typical SDLC frame format is given in Fig. 2.1. The opening and closing flags form the frame boundaries and all the fields within the frame are positionally related to these flags. Within the frame boundaries, SDLC is code transparent, i.e., the content and format of the data between the flags may take any form and be from any source. This, of course, implies that a

opening flag	address byte	control byte	info. bytes	crc bytes	closing flag
01111110	8 bits	8 bits	0 to n bits	16 bits	01111110

Fig.2(1) SDLC frame format

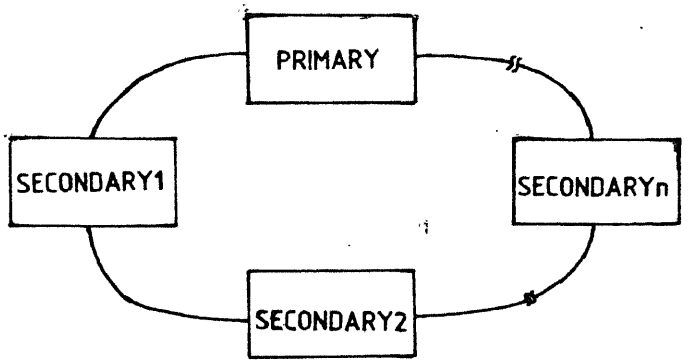


Fig.2(2) Basic LOOP Network

Table 1. Comparison of some LSI chips that support SDLC loop mode [Ref. No. 5]

	Intel 8274	Intel 8273	W.Dig. 1933	Signet. 2652	SMScor 5025	Motrola 6824
Speed	880K	64K	1.5M	1M/2M	2M	1M
Selective receive	✓	✓	✓	✓	✓	
General receive	✓	✓	✓	✓	✓	✓
All party addr.	✓	✓		✓	✓	
NRZI		✓	✓			✓
Internal PLL		✓				
EOP detect.		✓	✓	✓	✓	✓
Var. char. length	✓		✓	✓	✓	✓
Multiprotocol	✓			✓	✓	

flag pattern must never occur within a frame. This is ensured by the zero bit insertion/deletion (ZBI) technique. The transmitter/receiver inserts/deletes a binary 0 after any string of 5 contiguous binary 1s between the flags. ZBI along with non-return to zero invert (NRZI) mode of encoding (i.e., signal condition does not change for a transmitted binary 1 but changes for a binary 0) also ensures transitions in the signal line, thus aiding easy clock recovery [8].

2.1.2 Choice of Loop as the Topology

The basic loop network, as depicted in Fig. 2.2 consists of a controller station (primary) and one or more secondary nodes. Communications on the loop are controlled by the primary. Secondary stations operate in repeater mode. Secondary stations must be polled by the primary before they can respond. Communications in the loop are limited to that between primary and secondary. Secondaries cannot directly talk with each other.

One salient feature of the SDLC loop is the End-of-Poll (EOP) character. This consists of a binary 0 followed by 7 binary 1s (01111111). Each node can access the loop only after it 'captures' the EOP character or token. All secondaries normally keep on repeating all the incoming messages down the loop with some delay (typically 1 bit). When an EOP is received the secondary checks to see if it has any packet

to be transmitted. If it does, it changes the last bit (1) of EOP to a 0, thus making it a flag pattern, before forwarding it. Now the secondary has captured the token and hence the loop. It switches to transmit mode and inserts its own packet(s). Once the transmission is over, it switches back to repeater mode and the EOP pattern gets automatically forwarded [Note that in the standard SDLC loop, after transmitting its own packets primary keeps on sending contiguous binary 1s . The final 0 of the closing flag of the previous frame plus the repeated 1s from the primary form a new EOP for the next down-loop secondary]. If a secondary does not have any frame to be transmitted, it simply does not capture the loop. Once the EOP has been repeated, the node must wait for next EOP for transmitting its packets.

The greatest advantage of the loop mode SDLC is its low cost implementation. Loop mode SDLC, being the simplest requires less intelligence than a station on a multi-point or point-to-point network. The interconnection cost is also low for a loop type architecture [5].

In bus type of networks, like Ethernet, the need for collision detection places special requirements on the transmission medium and modems. Token passing loop networks do not require any collision detection and are therefore free from associated problems and complexities.

For the token passing type networks, the access time is deterministic, not contention-based. Therefore, response time is predictable for varying loads and the maximum message length for one token pass is known. The maximum time for one complete circulation of the token (in the absence of errors) is determined readily by

$$T = N\tau + \sum_{n=1}^{N} t_n \qquad (1)$$

where n = node number

N = number of nodes

τ = maximum holding time of token by any node

and t_n = the propagation time between the nodes

The maximum time between successive accesses by node N is then

$$T_N = (N-1)\tau + \sum_{n=1}^{N} t_n \qquad (2)$$

τ and N are network parameters that may be tailored to a particular application [7]. Thus token-passing networks are desirable in real time applications where fixed bounded response times are required. A typical case is that of process control environment.

2.1.3 Device Selection

An LSI implementation of data link controls allows on-chip inclusion of all the SDLC loop mode features, drastically reducing

hardware and software requirements.

Any such device chosen should atleast provide

(i) automatic zero bit insertion/deletion

(ii) automatic CRC generation and checking

and (iii) TTL compatibility.

Some of the available chips, with all the above facilities are compared in Table 1 in terms of other desirable features.

Intel 8273 is the only chip amongst these with internal synchronisation capability but is rather slow for our requirements. Intel 8274 has the major disadvantage of the absence of EOP detection capability. This calls for either external hardware circuitary or appropriate compromises in the system configuration. Compatibility with 8085 is an overweighing factor in favour of 8274, as compared to other chips. Again multi-protocol support with 2 independent channels is attractive, when one intends to extend the basic loop network to one with a Host-to-Host communication capability, through IMP-Host interfaces.

8274 MPSC is a versatile chip with various interface options like polled, wait, interrupt and DMA. Higher data rate requirements suggest the use of DMA mode of operation for data transfers in the loop. Intel's programmable DMA controller, 8237 is readily compatible with 8274. Further this has 4 independent DMA channels and features like memory to memory transfer

and software DMA requests. So an 8274-8237 configuration has been chosen for the basic loop implementation.

2.1.4 Clock Recovery Scheme

The synchronisation for the receive clock must be derived externally, from the receive data stream. Here the choice is between a Digital Phase Locked Loop type circuit or a pulse gulper type circuit. DPLL has the advantages of better stability and better performance, especially at high data rates. But the relatively inexpensive and simpler pulse gulper type circuit is expected to be satisfactory at the data rates under consideration.

2.1.5 Transmission Medium

Optical fibres (with data rates upto 100 Mbits/s), being very costly, do not suit our purpose of a low cost network. Coaxial cable has the advantages of physical ruggedness and better noise immunity compared to twisted pairs. But since our current attempts are for data rates around 200 KBs and reasonably small signal line lengths, the relatively cheap, twisted wire pair medium was used in our design.

2.2 OVERALL SET UP

The 8274's channel A is used for the IMP-to-IMP network

interface and channel B for IMP-to-Host interface. As mentioned earlier, the absence of EOP detection capability in 8274 calls for either additional hardware or appropriate compromises on the overall set up of the network. Currently the latter option is selected from cost considerations. We have a loop configuration in which each station receives each packet completely and then repeats it to the next down loop node. In other words, a store and repeat type scheme is attempted.

Each station has its RxD connected to TxD of the next uploop node and its TxD connected to RxD of next downloop node. Currently simple TTL inverting buffers are used to drive these signals. But in an actual practical situation, where lengths of signal lines are more, RS 422A type interfaces (using 3688/3689 or similar ICs) must be used. The IMP-to-IMP interface uses SDLC protocol, with transmission and reception by DMA.

External synchronisation circuitry ensures that receive clock is synchronised with respect to the receive data. Currently the maximum data rate attempted, for the loop, is 192 KBs.

[It may be noted here, that as far as possible no compromise that may hinder future upgradations of the system has been accommodated. This was the main motivation for choosing 8274-8237 combination, which can support data rates upto 880 KBs. Possible future upgradations include external

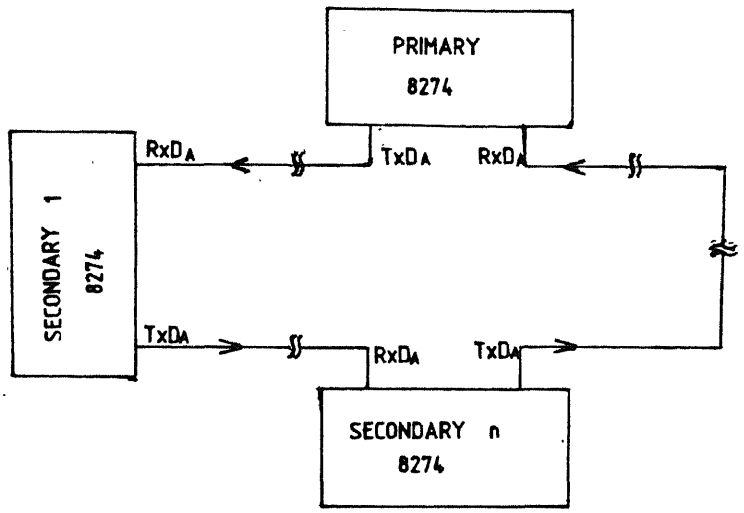


Fig.2(3) Network IMP - IMP interface

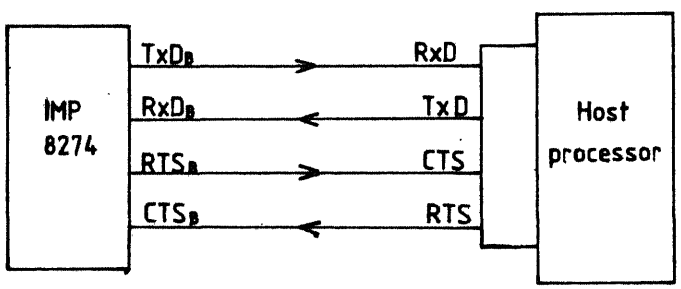


Fig.2(4) IMP - Host interface

hardware circuitries for EOP detection and, if necessary, a DPLL type clock recovery scheme. These two are relatively independent accessories, which may be added or enhanced at a later date, without changing the core design].

IMP-to-Host transactions are achieved via an RS-232 type interface using 8274 channel B. Most of the present day PCs and minicomputers usually provide an RS-232 type serial port. So the choice of RS-232 interface makes it possible to link these systems as host processors in the network.

By manipulating its own RTS, Host or IMP can control the clear-to-send line of the other. This ensures proper flow control between the Host and IMP. Receive and transmit clock for channel B of 8274 are provided from counter O of the 8253 timer, included in the set up.

8K byte ROM for storing program code and 6K byte RAM for buffering and maintaining pointers, variables etc. are also provided on the card. 2732A ROMs (with access time = 200 ns) and 6116 static RAMs (with access time = 120 ns) are chosen for this purpose. Since both of these conform to the JEDEC, 24 pin, 8 bit wide standard, the layouting of the memory section becomes easy.

CHAPTER 3

HARDWARE DETAILS

The actual hardware design aspects of the LAN interface card are discussed here. Wherever relevant, block schematics and timing diagrams have been used for illustration. Complete circuit diagrams are also included.

3.1 IITK WORKSTATION AND BUFFER CARD

The IITK 8085 based workstations have been made use of in the system development. In the preliminary stage, the intention was to have workstations connected together in a loop. Now with the Host-IMP RS-232 interface set up using 8274 channel B, the role of workstations has been reduced to that of a simple 8085 processor running networking IMP software. At this stage, one may, as well replace the workstation with an 8085 processor and minimal amount of hardware interface. This hardware, among other things, should take care of address/data demultiplexing. Branchings in the case of interrupts and RST instructions should also be handled appropriately.

The block schematic and the circuit diagram for the buffer card used is given in Fig. 3.1 and Fig. 3.2 respectively. This buffer card is used to interface the workstation with the LAN interface card.

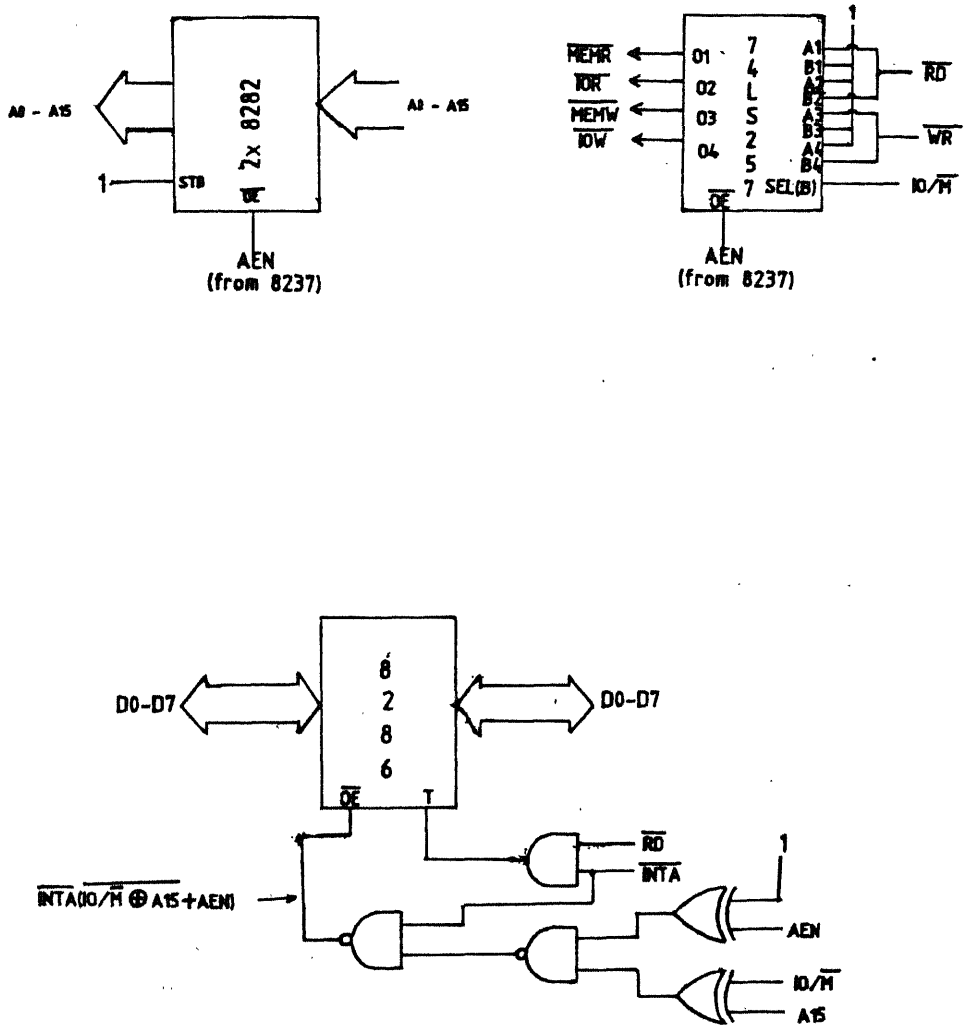


Fig.3(1) Block schematic for buffer card

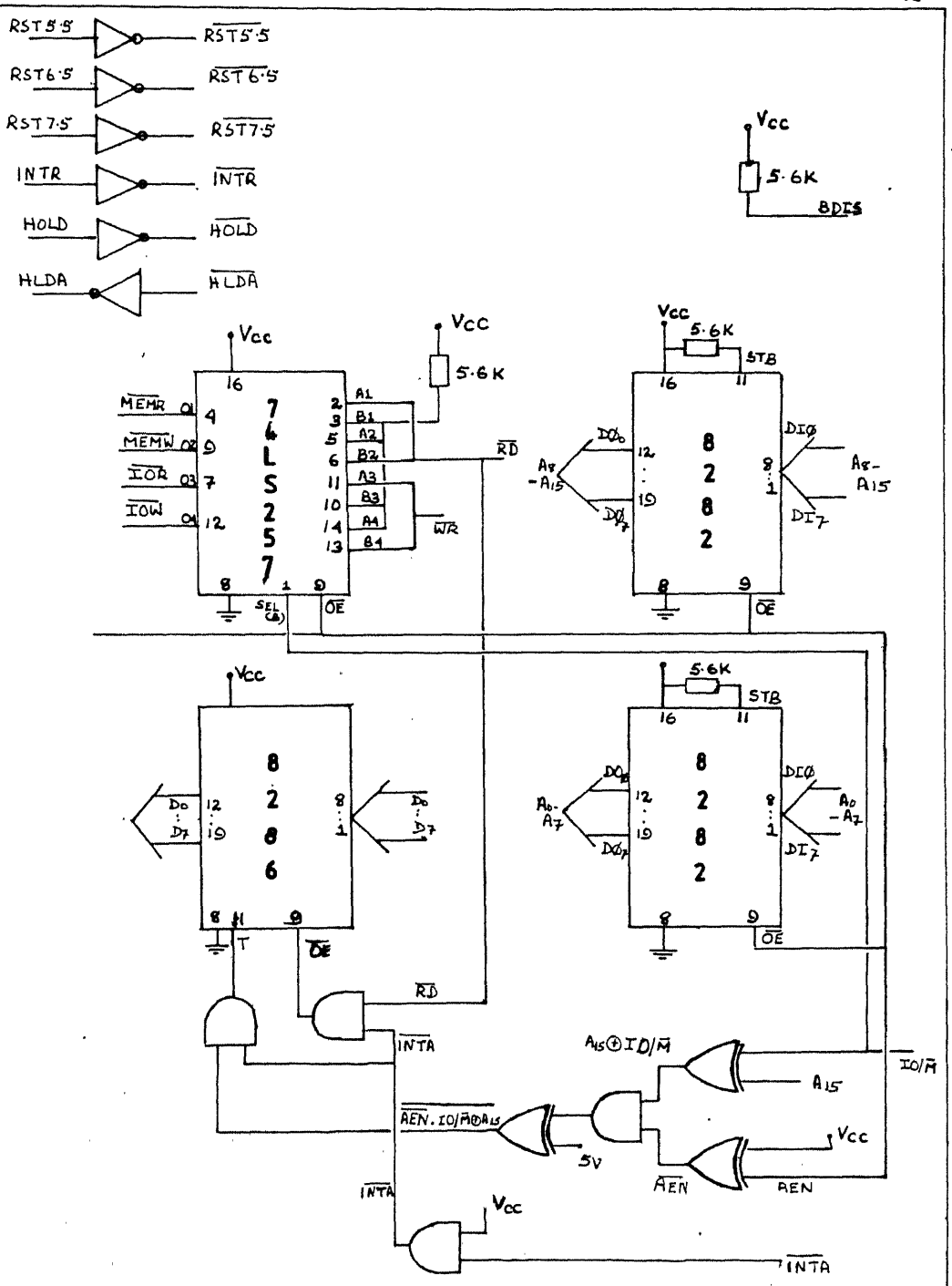


Fig.3(2) Buffer card circuit diagram

3.2 LAN INTERFACE CARD

3.2.1 Address Decoding

The address map for the card is given in Table 2. The card occupies addresses from 8000-BFFF (hex) and 00 - 3F (hex). 74LS138 decoders are used for address decoding and generating $\overline{CS}$ signals for the various devices. The details of the decoding logic are given in Fig. 3.3.

3.2.2 Memory

8K ROM area and 6K RAM area are provided in the card. 8K ROM area consists of two 2732A ROMS, each with 4K x 8 byte capacity. Three 6116 RAMs, each with 2Kx8 capacity, constitutes the 6K RAM area. Note that these devices with access times 200 ns (2732A) and 120 ns (6116) are satisfactory for data rates upto 880 KBs.

3.2.3 Clock Recovery Circuit

The pulse gulper type clock recovery circuit, used in the set up, is depicted in Fig. 3.6. This basically involves an 'edge' extraction logic, a divide by 16 counter and a mono-stable multi. triggered by the 'edge'. The output of the mono-stable controls the clear input of the divide by 16 counter. So at each rising edge (see timing diagram in Fig. 3.7) the divide by 16 counter is cleared and the clock gets synchronized to incoming data stream.

Table 2. Address map for LAN interface card

		IO/$\overline{M}$	A15	A14	A13	A12	A11	
ROM	2732(1)	0	1	0	0	0	X	8000-8FFF
	2732(2)	0	1	0	0	1	X	9000-9FFF
RAM	6116(1)	0	1	0	1	0	0	A000-A7FF
	6116(2)	0	1	0	1	0	1	A800-AFFF
	6116(3)	0	1	0	1	1	0	B000-B7FF
Memory mapped I/O	8253		1	0	1	1	1	$B800_0$-B803
I/O mapped I/O	8237	1	0	0	0	0	X	00 - 0F
	8274	1	0	0	0	1	1	18 - 1B

Card occupies 8000 - BFFF (mem) and 00 - 3F (I/O)

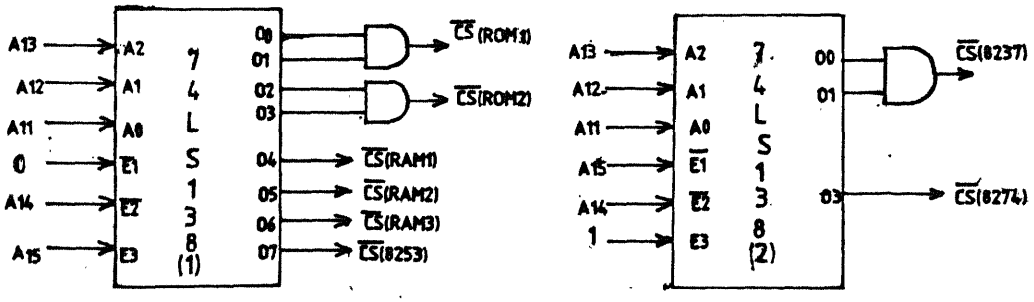


Fig.3(3) Decoding logic for LAN interface card

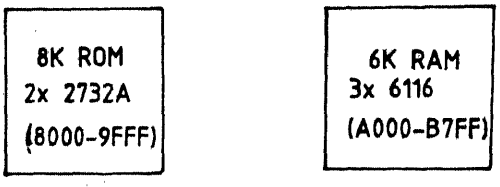


Fig.3(4) Memory Organisation

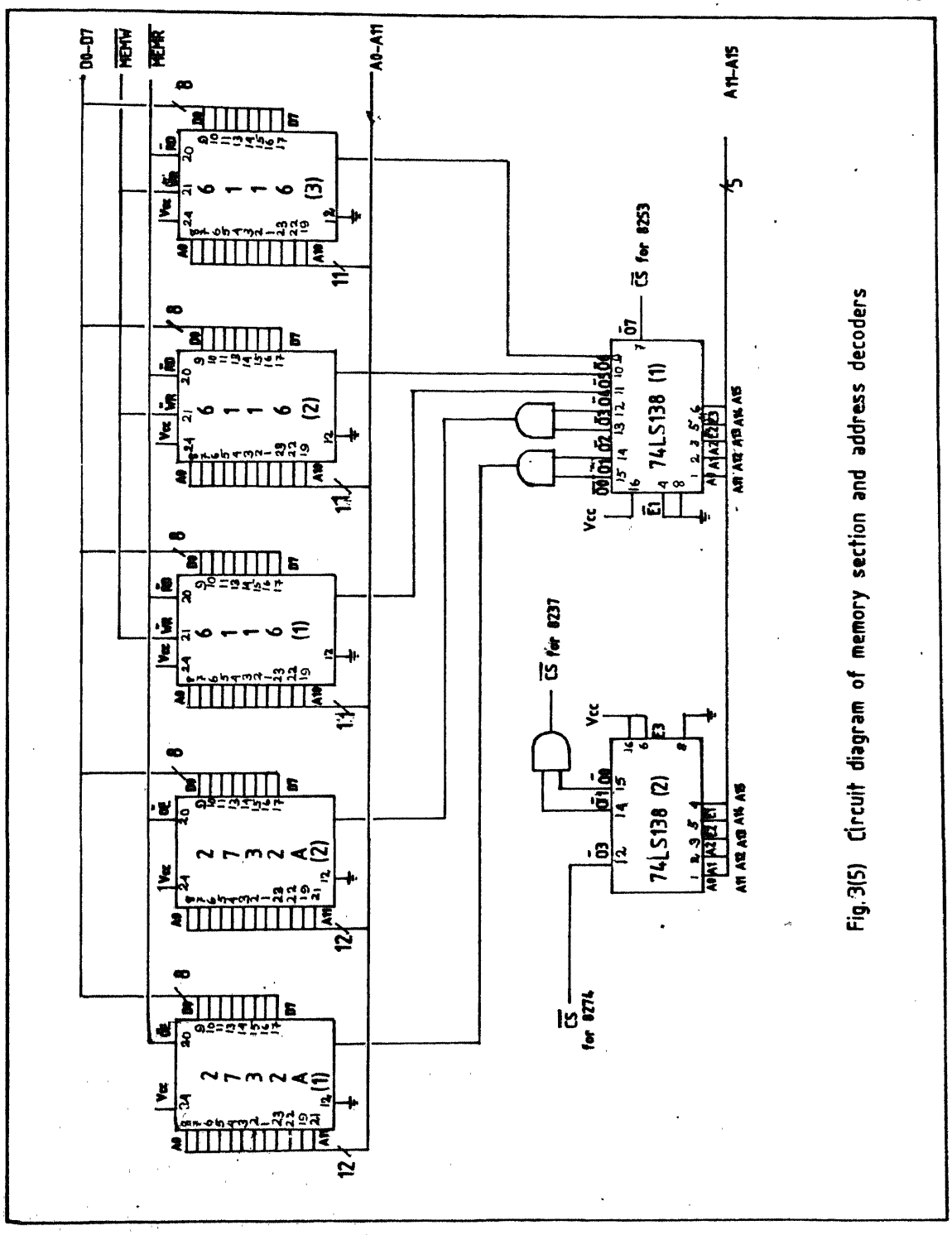


Fig.3(5) Circuit diagram of memory section and address decoders

7493 ICs are being used as divide by 16 counters and dual monostable multi 74123 for deriving narrow pulses from rising edges of RxD. For extracting the edge, the inherent propagation delay (typically 2 ns-10 ns) of three 74 series inverters is utilised. For the cases where this delay seems to be less than enough, a small capacitance (typically 10-15 pf) may be used. This connected across the output and ground of first inverter increases the propagation delay. Monostable multi periods have been set by external resister and capacitor (7.5 K and 22 pf). The clock 0 input of 7493 is directly connected to 8085 processor clock out. So we have a maximum data rate of 192 KBs with 3 MHz processor clock.

3.2.4 8253 Timer Interfacing

Processor clock out (3 MHz) is divided by 2 (using 7493) and fed to all the clock inputs of 8253. Gates of all the 3 counters are permanently pulled to Vcc through pull up resistors.

The output of counter 0 provides x16 baud clock (both TxC and RxC) for 8274 channel B. The other two counters currently unused, are to be used for time out purposes, in interrupt mode.

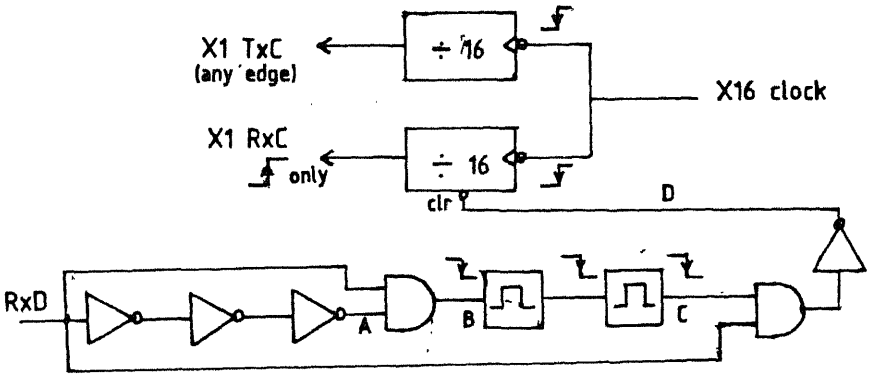


Fig.3(6) pulse gulper type clock synchronisation

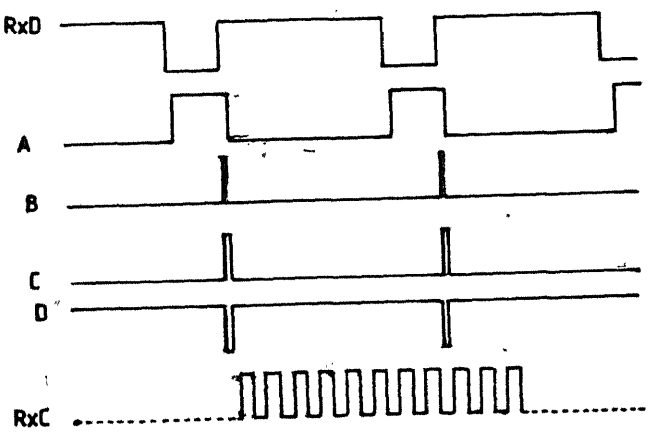


Fig.3(7) Timing diagram for synchronisaion circuit

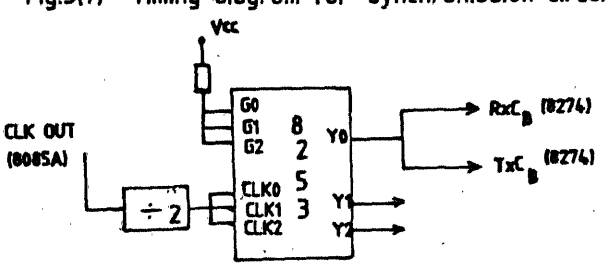


Fig.3(8) 8253 Timer interface

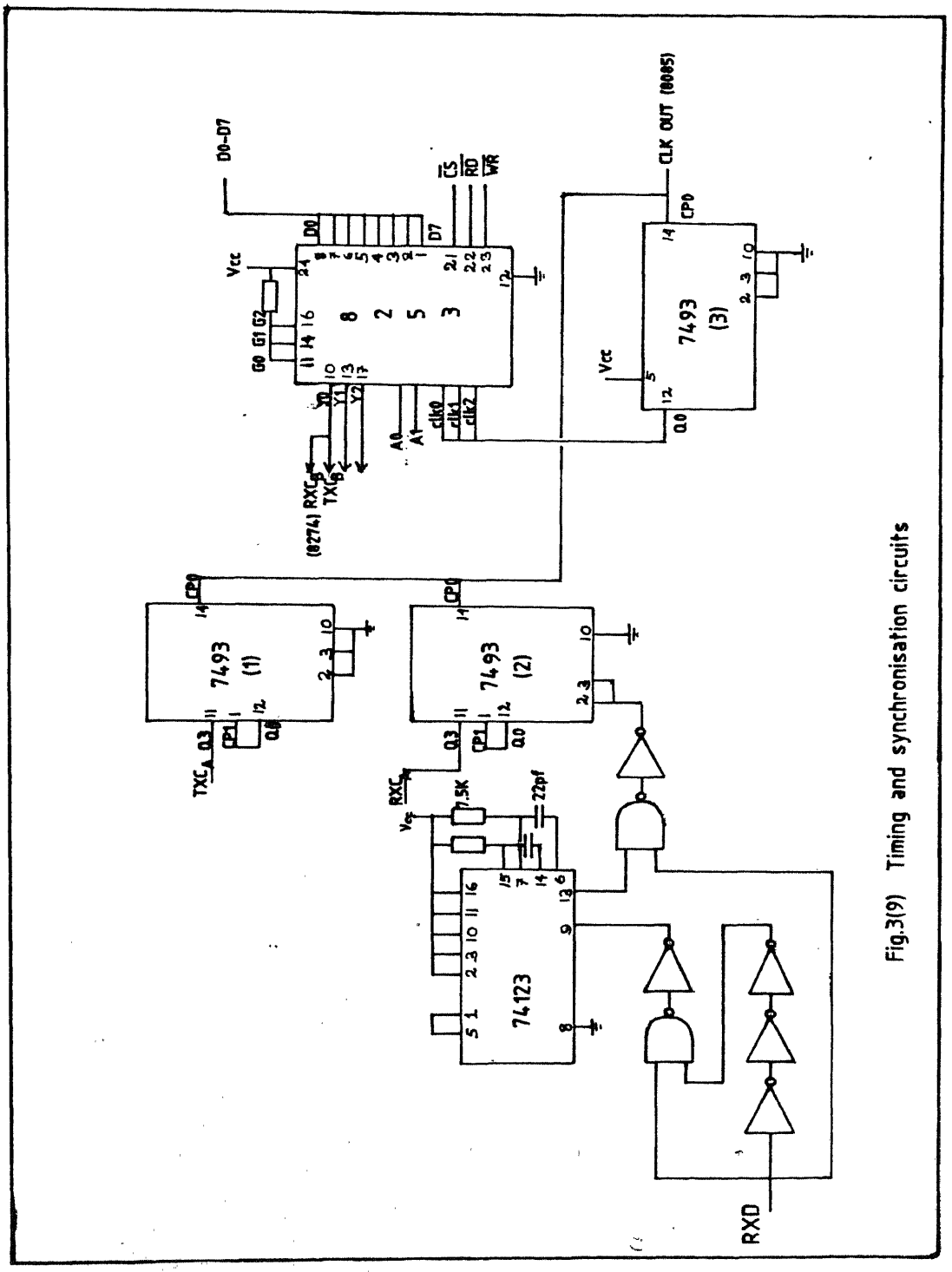


Fig.3(9) Timing and synchronisation circuits

3.2.5 8237 DMA Controller and DMA Acknowledgement Circuit

The configuration of 8237 interfacing is shown in Fig. 3.10. Octal latch 8282 has been used for latching onto data bus, the higher order address byte during DMA transfer. ADSTB and AEN lines from 8237 controls this latching operation. AEN when high, disables all the data and address lines to processor (see Fig. 3.1 Buffer card organisation). During DMA transfer, 8237 takes over the bus by making AEN high. 8274 channel A TxDRQ and RxDRQ drive the DRQO and DRQ1 lines of 8237 to initiate DMA action, whenever required. The associated DMA acknowledge circuit is given in Fig. 3.11.

3.2.6 8274 Interfacing

8274-MPSC channel A operates in SDLC, DMA mode and channel B in asynchronous, polled/interrupt mode. Channel A provides the network IMP-to-IMP interface and channel B, the IMP-to-Host interface.

RxD_A is connected to preceding node's TxD_A and TxD_A connected to succeeding node's RxD_A through appropriate line drivers. RxC and TxC are fed from divide by 16 counters using 7493 counters. Pulse gulper type clock recovery circuit derives the synchronisation for RxC from the received data stream. Receive and Transmit DMA request lines for channel A are connected to DRQO and DRQ1 respectively of 8237.

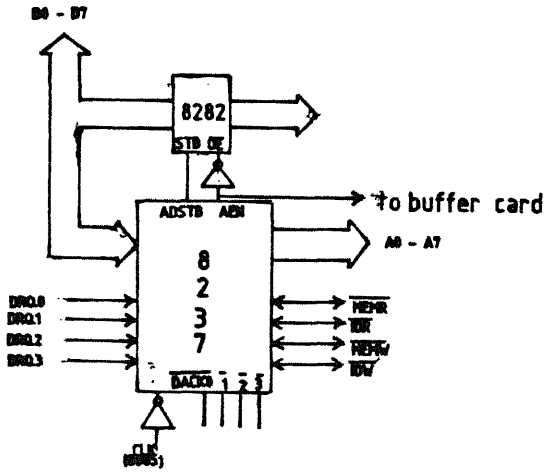


Fig.3(10) 8237 DMA controller interface

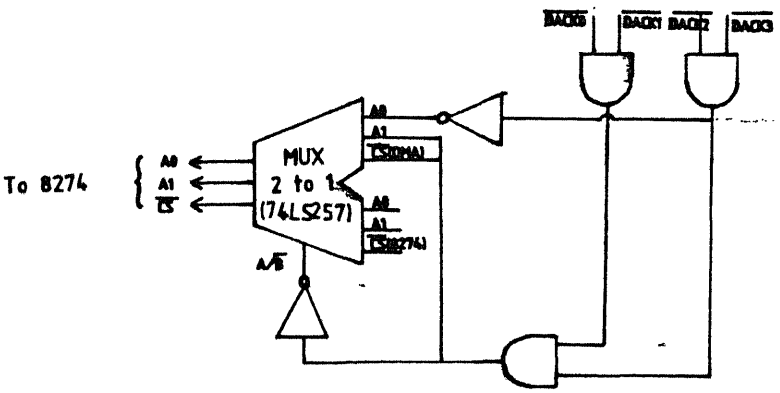


Fig.3(11) DMA acknowledge circuit

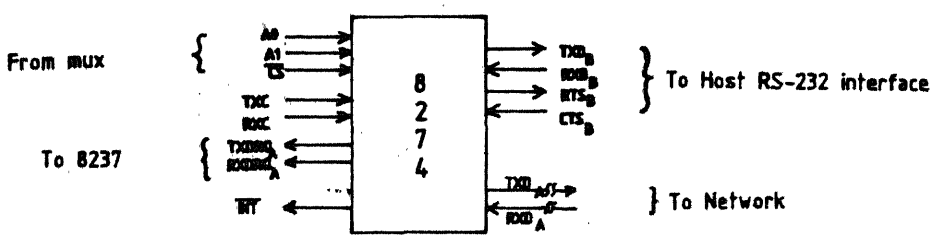


Fig.3(12) 8274 MPSC interface

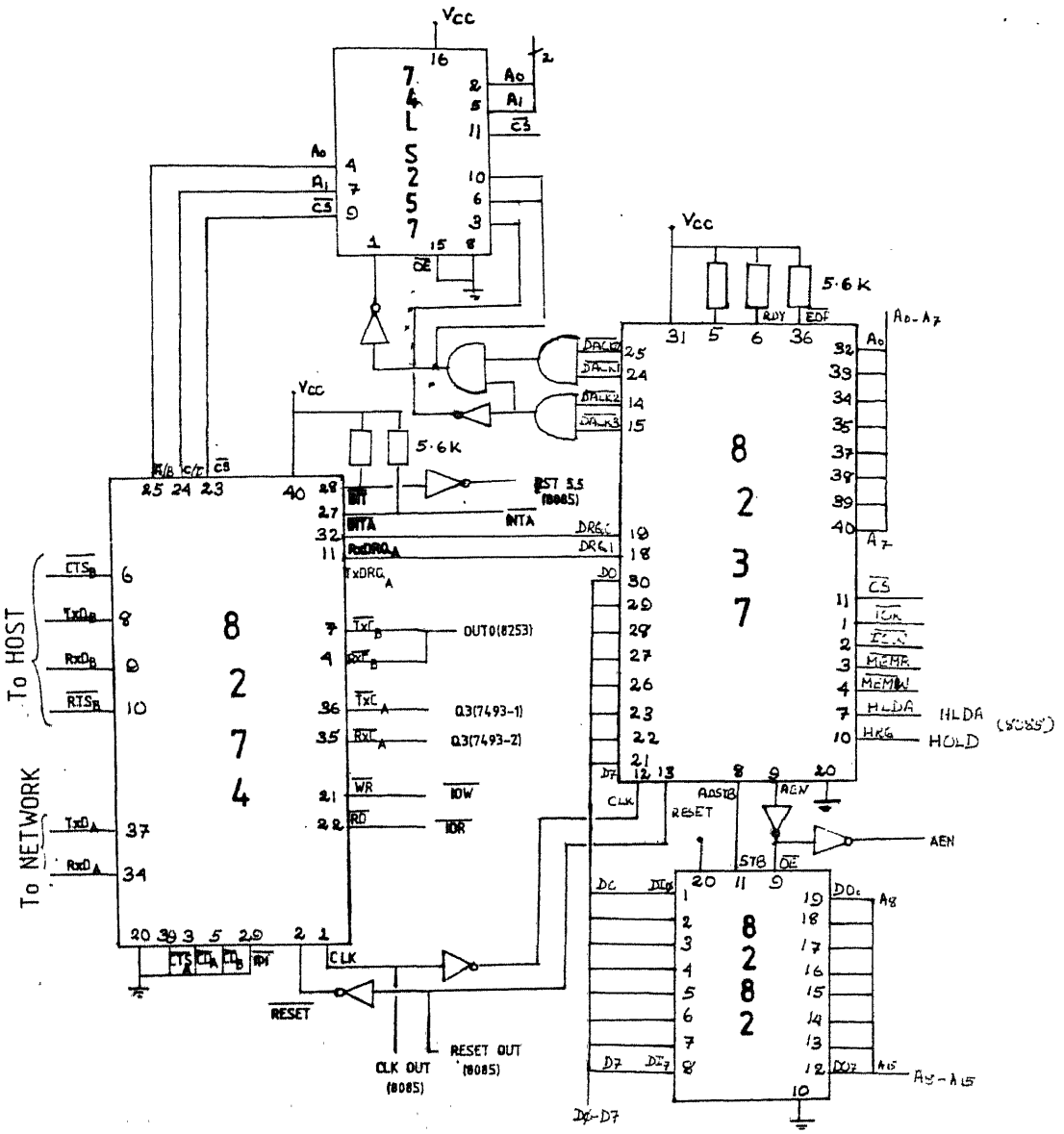


Fig.3(13) 8274 8237 & assosciated circuitry

Open collector output $\overline{INT}$ of 8274 is pulled up through a 5.6K resistor and is connected to RST 5.5 interrupt line of processor after inversion. $\overline{INTA}$ line is also pulled high and connected to processor $\overline{INTA}$ line. This is done, so that later on, one may switch over to vectored mode of interrupts from 8274, with minor changes in software. Since we don't require to connect more than one MPSC at one node, $\overline{IPI}$ line is grounded to avoid unexpected behaviour of MPSC.

8274 channel B, programmed in asynchronous mode with 2 stop bits, even parity and x16 clock, provides the IMP-to-Host interface. RxD_B of 8274 connected to TxD of Host, TxD_B to RxD, CTS_B to RTS and RTS_B to CTS (see Fig. 2.4) through 1488/1489 inverting buffers. By manipulating one's own RTS, IMP and Host can thus control the CTS line of the other and hence the data flow from the other.

3.3 TRANSMISSION AND RECEPTION

During transmission of a frame, the CPU first initialises and sets up the DMA channel. Then the first character of the frame is loaded by CPU to the transmit data buffer of 8274 channel A. Character in the transmit data buffer, gets automatically loaded to transmit serial shift register. The bits in serial shift register is shifted out through a 2-bit delay onto the Tx Data line on the falling edge of the transmit clock. This

2-bit delay is used to synchronise the internal shift clock with external transmit clock. The data in the shift register is also presented to the zero bit insertion logic which inserts a 0 after sensing five contiguous 1s in the data stream. Last 3 bits of the 20 bit serial shift register indicate to the internal control logic that the current data byte has been shifted out of the register and next byte, if any, in transmit data buffer gets loaded to shift register. Each time the character in Tx data buffer gets loaded to serial shift register, the transmit data buffer becomes empty. This generates a DMA request (Tx DRQ) and the next byte gets loaded to Tx Data buffer by DMA. DMA action progresses without any CPU intervention. In parallel to all these activity, the CRC-generator is computing the CRC on the bits shifted out through serial shift register. When the DMA controller reaches the terminal count, it will not respond to the DMA request. This results in both Tx Data Buffer and serial shift register becoming empty and transmit underrun condition occurs. CRC bytes now get loaded to shift register (provided Tx underrun/EOM latch is reset by CPU during the transmission) and get transmitted. After this again DMA request is generated and MPSC underruns. Now that the Tx underrun/EOM latch is set, flags get loaded into shift register and get transmitted. Each time MPSC underruns, an external status interrupt also is generated which is reset by Reset External Status Interrupt command.

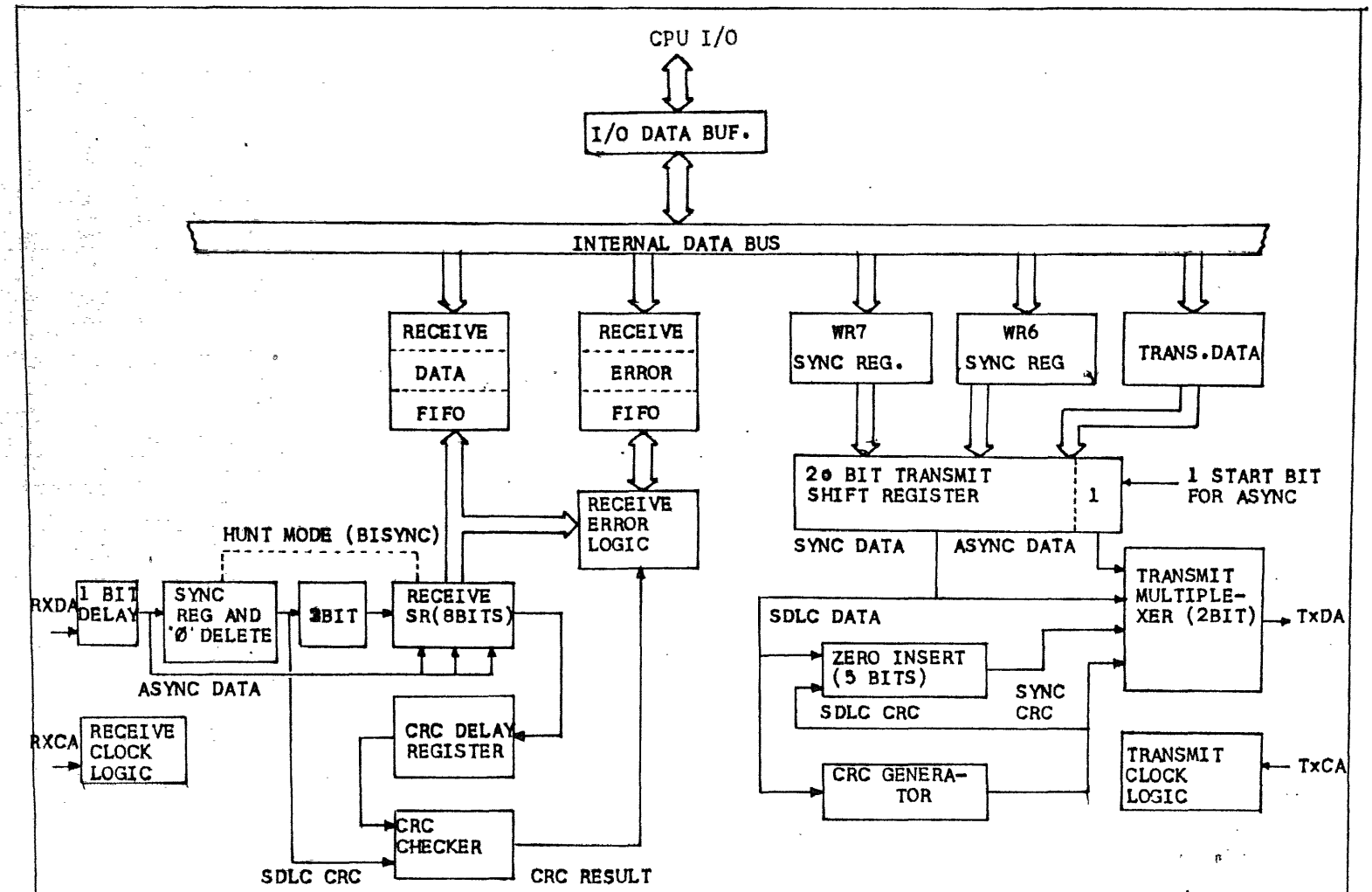
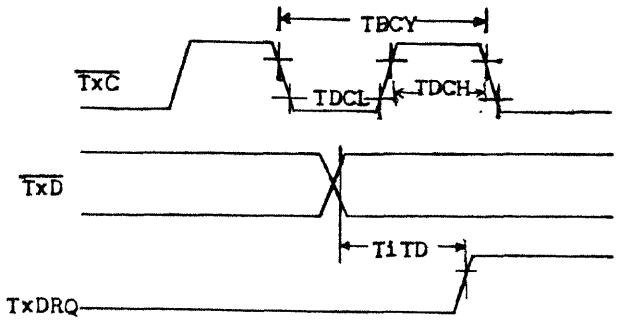


Fig. 3(14) TRANSMIT AND RECEIVE DATA PATH (8274) [Ref. No. 9]

Receive data line is sampled at mid bit time, on the rising edge of RxC. The received data is passed through a one bit delay before it is presented for flag comparison. After this, the incoming data is passed through the synch. register where data pattern is continuosuly monitored for contiguous 1s for zero deletion logic. The data then enters the 3 bit buffer and the receive shift register. From the receive register, the data is transferred to the 3 byte deep FIFO. The data is transferred to the top of the FIFO at chip clock rate (not receive clock). The 3 bit deep receive error FIFO indicates any error condition that must have occurred during reception of a frame. Note that the receive DMA channel is initialised and left enabled prior to reception. During a frame reception, after each character is completely assembled at MPSC, a receive DMA request is generated. The received data gets transferred to memory under DMA control. Since MPSC is programmed in interrupt on first receive character mode, first character of a frame generates a Receive interrupt also. This is simply ignored. The reception proceeds without any CPU intervention. Now when the MPSC receives the closing flag denoting the end-of-frame (EOF) a special received interrupt occurs. Any receive error condition also will generate a special received interrupt. In both cases, the Rx DMA request gets disabled and reception stops. While all this is happening, the CRC checker is checking the CRC on the

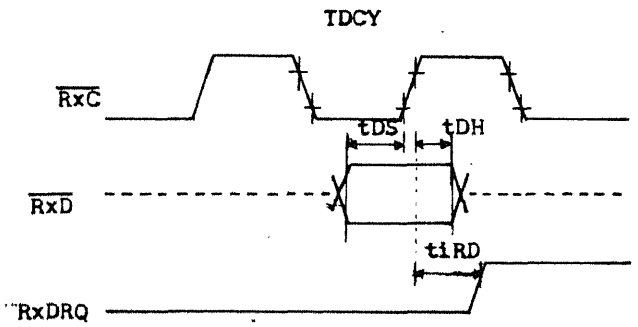
incoming data. The computed CRC is checked with the CRC bytes attached to incoming data and an error generated under a no-check condition. In the special received interrupt service routine, one may check whether any error has occurred during reception and if so reject the frame. Otherwise the packet is accepted. The special received interrupt is reset by an Error Reset command.



TDCY- CLOCK CYCLE TIME
TDCH - CLOCK HIGH TIME
TDCL - CLOCK LOW TIME

TiTD - $\overline{TxC}$ to $\overline{INT}$ (TxDRQ) delay

Fig. 3(15a) Transmit Data Cycle



TDH - RxD Hold after $\overline{RxC}$

TDS - RxD set up to $\overline{RxC}$

TiRD - $\overline{RxC}$ to $\overline{INT}$ (RxDRQ) delay

Fig. 3(15b) Receive Data Cycle

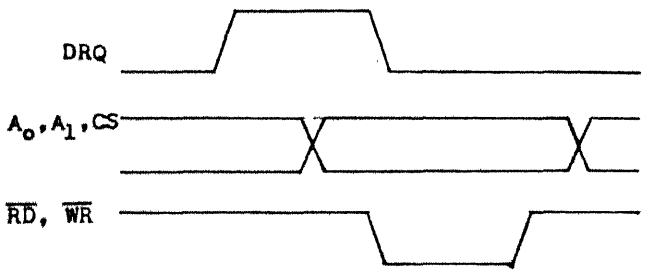


Fig. 3(16) DMA Timing

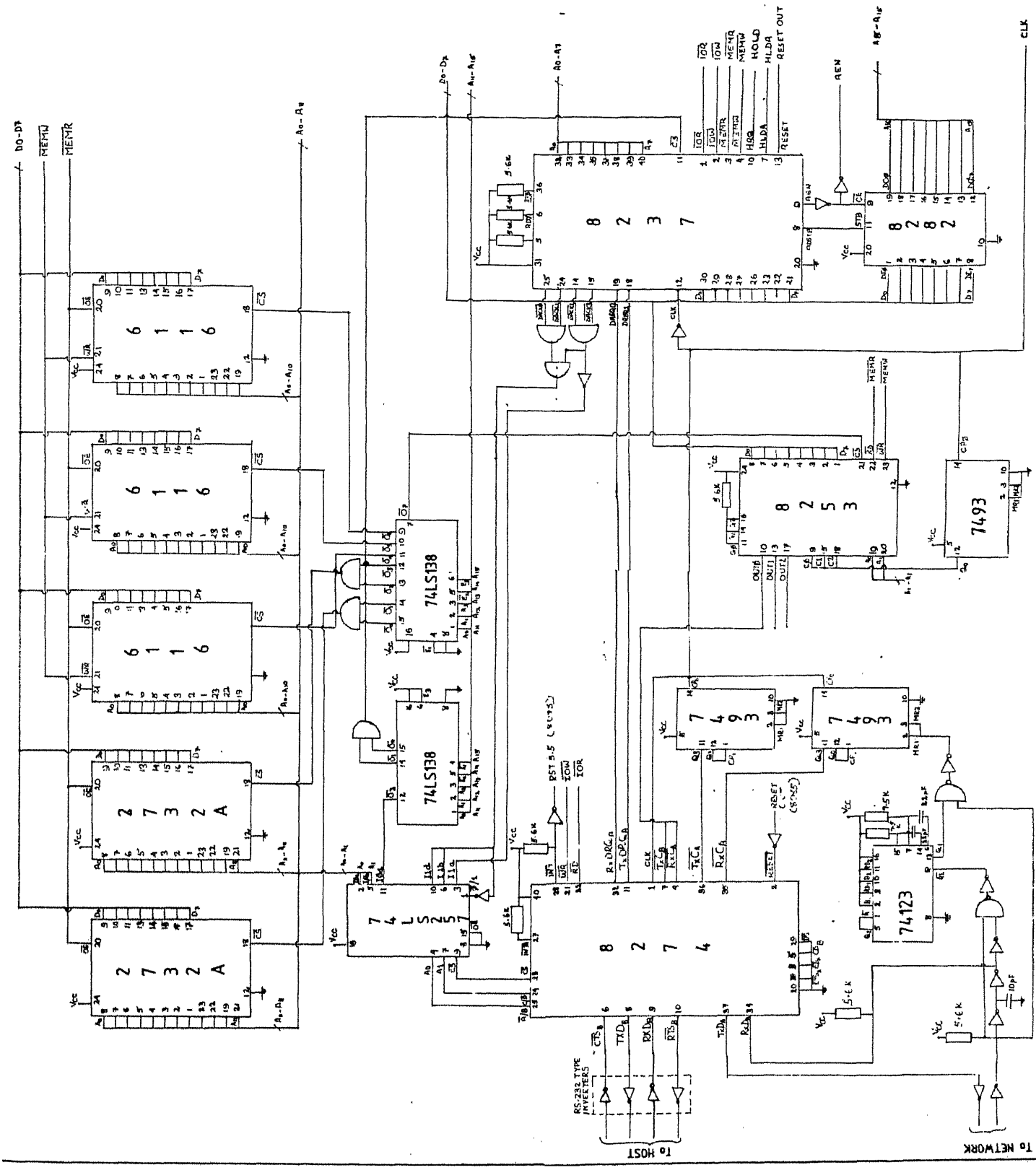


Fig.3(17) Complete circuit diagram —LAN Interface card

CHAPTER 4

APPLICATION SOFTWARE

The program development for the implementation was done in 8085 assembly language, using the facilities of Intel's Intellec MDS-Series III. Certain general considerations in the application software development are discussed here. Also a brief outline of the algorithms and database , highlighting the salient features are included. Detailed description of database organisation and algorithm are available in Appendices A and B.

4.1 SOME GENERAL CONSIDERATIONS

4.1.1 Need to have a Circulating Token

The communications in the loop rely on the fact that secondaries repeat all the packets received. But each node should know when to transmit its own packet or packets. So a token, the possession of which gives the right to transmit, is passed around the loop. Each node keeps on repeating all the packets until the token or EOP is received. Once EOP is received, the node changes from repeat mode to transmit mode. It checks to see if it has any packets, message or acknowledgements, to transmit. If so these are transmitted followed by EOP.

If no packets are to be transmitted, EOP is simply forwarded. After transmission of EOP, the node switches back to repeat mode and remains in this mode, until next EOP reception.

Since only the node which has the token can transmit and there is only one token, collisions or contentions do not occur. Again, all nodes get a chance to transmit as the token moves around the loop. This ensures fairness of access. It may be noted here, that any situation in which token gets lost results in the loop to hang up. This calls for a token time out and recovery scheme at primary. [In this context, the remarks in Ref. [6] about locating faults in a ring communication system are interesting. It suggests a token time out scheme at each node. The node which times out first, generates an error message, pinning down the fault to immediately preceding node].

The fact that each packet is completely received before it is transmitted, in the present implementation and full duplex capability of 8274 indicates a possibility of a store and forward type network also. Each station keeps on repeating all packets it receives. It may transmit its own packet, any time, provided there is no packet to be repeated. This option was also tried out in the preliminary stage and found to be working satisfactorily.

4.1.2 Choice of ARQ Scheme

As is common for most of LANS, the stop and wait ARQ scheme, which demands least complexities in algorithms and buffer management has been chosen. In this scheme, each node transmits a packet to another node, only after the acknowledgement to the previous packet, if any, sent to latter is received. So proper sequencing of packets is automatically ensured.

4.1.3 Role of Primary

Network, being an interconnected collection of autonomous computers, as mentioned previously, one would like to avoid any master/slave relationships between the nodes. But certain factors, discussed below, necessitates the presence of a 'primary' node, with a special status.

One of the functions of this special node is to remove those packets which have already finished circulation around the loop. Of course, we can and do have a scheme in which each node takes out packets addressed to it. But this does not take care of broadcast packets and packets addressed to nodes, not currently in the loop. It is primary's function to remove these packets. But at the same time, it should be ensured that the packet has reached its destination(s) before it is taken out. So primary, when it receives a packet retransmits it once around the loop and then takes it out. To avoid duplications,

broadcast packets are received by secondaries only after they are retransmitted from primary, i.e., in their second leg.

In the beginning or during a recovery, somebody has to initiate the networking action by generating the EOP. This also is done by primary.

Packets that are circulating around the loop are buffered by primary until it receives the EOP pattern back. Once EOP is received, it repeats those packets which are to be repeated and inputs to its own receive buffer those packets which are to be input. Note that here also primary differs from the secondary node, which immediately repeats the packets received from the loop.

Apart from the above differences, the primary node is very much like the other nodes.

4.2 DATABASE ORGANISATION

The database consists of various buffers, pointers, status bytes and other variables. Buffers are used to store and retain packets to be repeated, input or output at the node. Pointers and variables are meant for proper management of these buffers. Status bytes give correct information regarding the status of the other nodes in the loop and one's own status.

Circular buffer, receive buffers and transmit buffers constitute buffer area for the implementation. The circular

buffer is meant for receiving the incoming packets from the network. It is of length 1K byte for secondary and 2K byte for primary. A larger circular buffer has been set for primary since it has to retain all received packets until EOP is received. As the number of nodes in the loop increases, one may have to have a still larger circular buffer at the primary. Four receive buffers of length 263 bytes each retain packets to be output to user. Of the two transmit buffers of length 261 bytes each, one will be designated for transmit to network and one for input from user side. So while a packet is getting transmitted, the user can input to the other transmit buffer. Other than these, we also have an acknowledgement buffer (32+1 byte) and start address circular buffer. Acknowledgement buffer gives the number of acknowledgements pending and the nodes to which these are pending. Start address buffer gives the starting address of the packets stored in the circular buffer. At a time, we can retain atmost seven packets in the circular buffer.

A 256 byte user table, giving the status of other nodes and state of transactionswith them provides all relevant information for the basic loop action. Status about one's own node is available in status bytes ROUTE and TXSTAT. Other than these, there are also other buffer pointers and variables, to aid easy management of the buffers and smooth working of the loop.

The detailed description of these are available in Appendix A. It may be noted that this database organisation supports easy development and implementation of the networking algorithm .

4.3 NETWORKING ALGORITHM

4.3.1 Initialisation

On initialisation, the 8253, 8274 and 8237 are appropriately programmed. All buffers marked empty and all status bytes and pointers set to initial conditions. Jump vectors are written to ensure proper branching on interrupts.

The 8253 Timer counter 0 is programmed in mode 3 to generate x 16 baud clock for IMP-Host interface. The other two counters are programmed in interrupt on terminal count and left without loading the count value. This leaves the outputs at low state. These counters which are currently unused may be used for time out purposes.

The 8274 is programmed in status affect, non-vectored mode of interrupts. In status affect mode, the interrupt status vector automatically gets modified according to the cause of interrupt. This ensures faster response, by eliminating the need to poll the status and find out the cause of interrupt. As appropriate for the requirements, the 8274 channel is programmed in SDLC, DMA mode and channel B in asynchronous polled

mode. Channel A receives and transmits 8 bits per character without parity. Channel B transmits and receives 7 bits per character and a parity bit (even parity) other than the start bit and two stop bits.

The 8237 is initialised in single transfer mode of transmit and receive DMA channels. Receive DMA channel is left enabled prior to reception, to take care of high data rates. All the unused registers of 8237 are also loaded with dummy values, to avoid any unexpected behaviour.

After the initialisation is over, each node waits for a finite time, for other nodes also to get initialised. Then all the pending interrupts at the 8274 are reset and the interrupt system is enabled. Now the nodes begin to execute the networking algorithm. Primary forwards the EOP packet before it goes into master loop for networking. This is done after a sufficient delay after initialisation, (almost double that of the delay at secondaries), to ensure all other nodes are already executing networking software.

4.3.2 Master Loop

Master loop serves three functions :

i) inputting from user side

ii) outputting to user side

and iii) proper designation of the transmit buffers for transmit to network and receive from user.

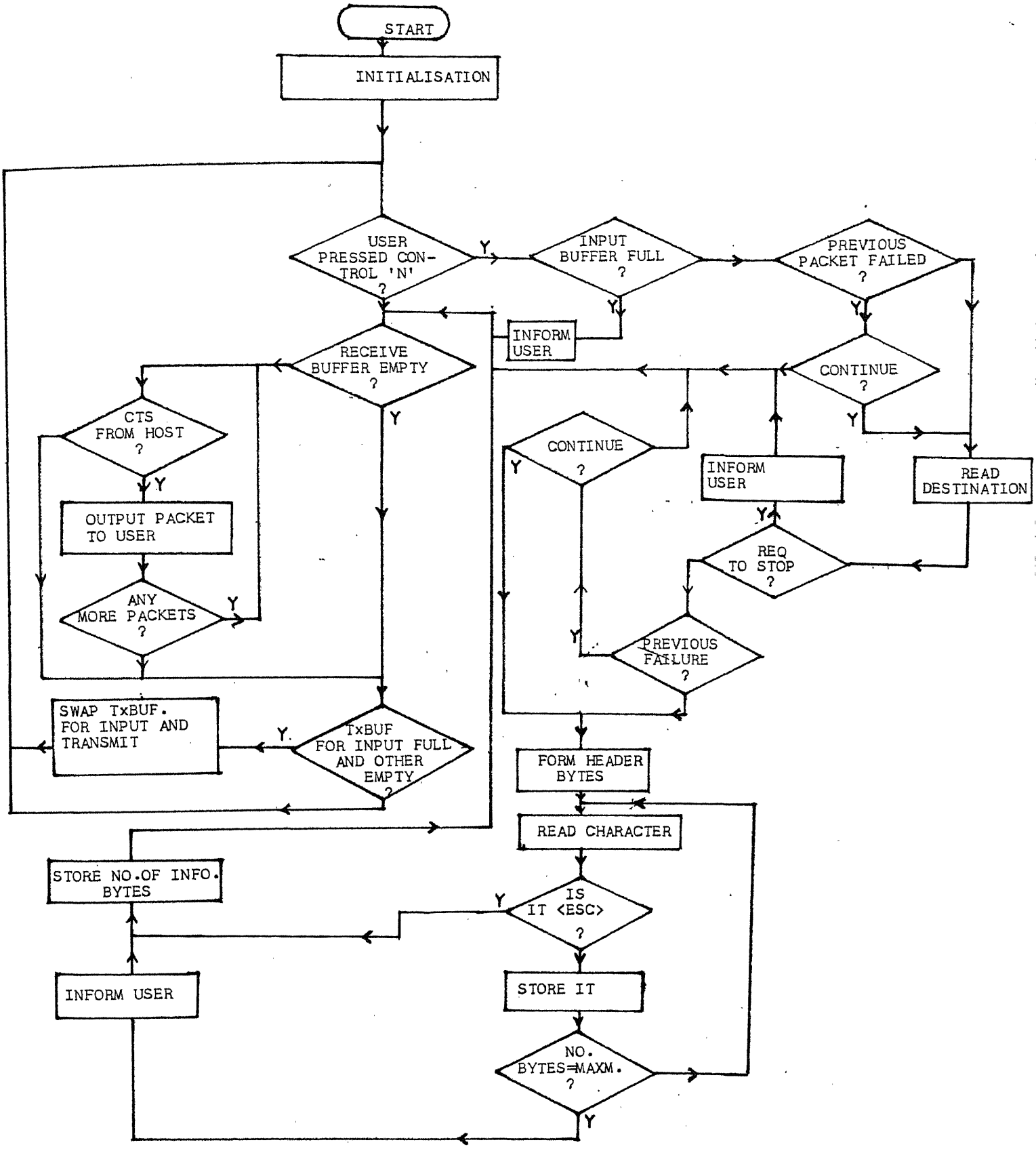


Fig.4(1) Flow chart-MASTERLOOP

The user invokes the input operation by typing '^N' . He is allowed to input only if the transmit buffer, designated for it, is empty. Otherwise user is informed appropriately and inputting is aborted. During inputting, user will be asked to type in destination followed by the message. In the event of a previous transmission failure or the previous failure of transmission to current destination, the user is informed and queried whether to continue. Inputting may be aborted or continued by user here as he desires. If the destination has requested for stopping all transmissions to it, then also the inputting is aborted notifying the user about this. Inputting of the message may be terminated by the user by typing <ESC>. Otherwise, system by itself terminates inputting when maximum allowed length is reached, informing the user of this. This user friendly interactive approach is one of the salient features of the implementation.

User outputting will be done only if Host is asserting a clear-to-send. This gives the Host, freedom to execute its own programs without any interruption from loop, by keeping CTS to IMP low. Before the actual message is output, details like source, length etc. of the packet are also displayed for the benefit of the user.

At primary, circular buffer processing and transmit loop are also executed in master loop, after EOP is received.

4.3.3 Transmission and Reception

Transmissions and receptions in the loop are by DMA and takes place without much of CPU intervention. During transmission CPU sets up the channel and loads the first character. After this packet transmission proceeds under DMA control. To ensure transmission of CRC bytes, CPU also has to reset the Tx under run/EOM latch after loading first character. Prior to reception, the receive DMA channel is set up and left enabled. Here again reception of first character initiates DMA action, whereafter it proceeds under DMA control until closing flag is received. Initialisation for next reception is done by CPU in the special received interrupt service routine.

4.3.4 Interrupt Servicing

The only normally expected interrupts for channel A during networking action are :

(i) Special Received Interrupt
and (ii) External Status Interrupt.

Any other interrupts are simply reset and ignored.

On getting an interrupt CPU must read the interrupt status vector in RR2 to find out the cause of interrupt. In order to do so, first a pointer to status register RR2 is specified and then the status read from RR2. It may be noted that after

specifying the pointer, CPU must read the status register RR2. Otherwise no new interrupt requests will be accepted internally. Again any interrupt will result in an internal in service latch in 8274 getting set. This should be reset at the end of each ISR by an End of Interrupt command. Otherwise no lower priority interrupts will be accepted thereafter.

(i) Special Received Interrupt

Special received interrupt occurs during a reception due to (i) reception of the closing flag (end of frame), or (ii) any receive error condition like receive overrun.

When this interrupt occurs, the receive DMA request is disabled and the reception stops. Preliminary processing of the currently received packet is done in this ISR. The packet is rejected if :

(a) receive overrun error occurred, or
(b) CRC error detected after end-of-frame, or
(c) circular buffer already contains seven packets, or
(d) number of free locations in the circular buffer after the reception of current packet is less than nine i.e., the length of EOP packet.

Correctly received packet with no CRC error is checked to see if it is an EOP packet. If so EOP received status bit (R7) is set and the packet rejected. Primary also rejects the

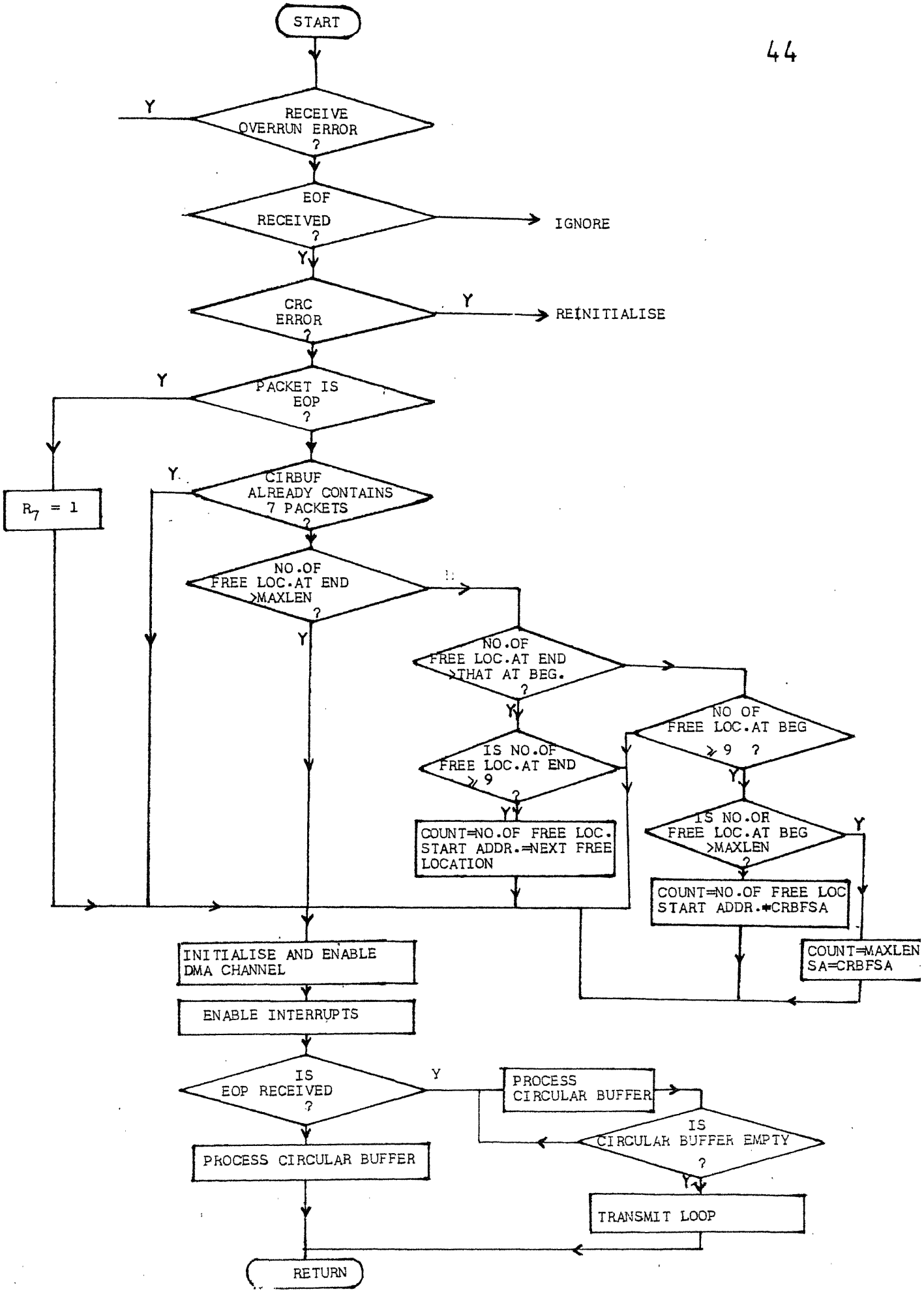


Fig.4(2) Flow chart-SPECIAL RECEIVED ISR

retransmitted packets (i.e., those with C7 = 1). Retransmission status bit (C7) of other packets are set and retained in circular buffer for later retransmission.

Reinitialisation of the receive DMA channel for next reception is also done in the special received ISR. Special received interrupt is reset by an error reset command.

Now before returning to the master loop, the secondary nodes check whether EOP has been received. If so processing of circular buffer until it is empty and transmit loop are executed. Otherwise the first packet, if any, in the circular buffer is processed. In the case of primary, processing of circular buffer and transmit loop are exeucted in master loop after reception of the EOP.

(ii) External Status Interrupt

In the event of an external status interrupt due to abort detection, that particular IMP and thereby the entire loop is reinitialised (see Section 4.4 - Errors and Reinitialisation). Transmission status bit (T5) is reset if the interrupt is due to transmit under run conditions, marking the completion of a transmission. Any other causes are simply ignored. Reset external status interrupt command resets this interrupt.

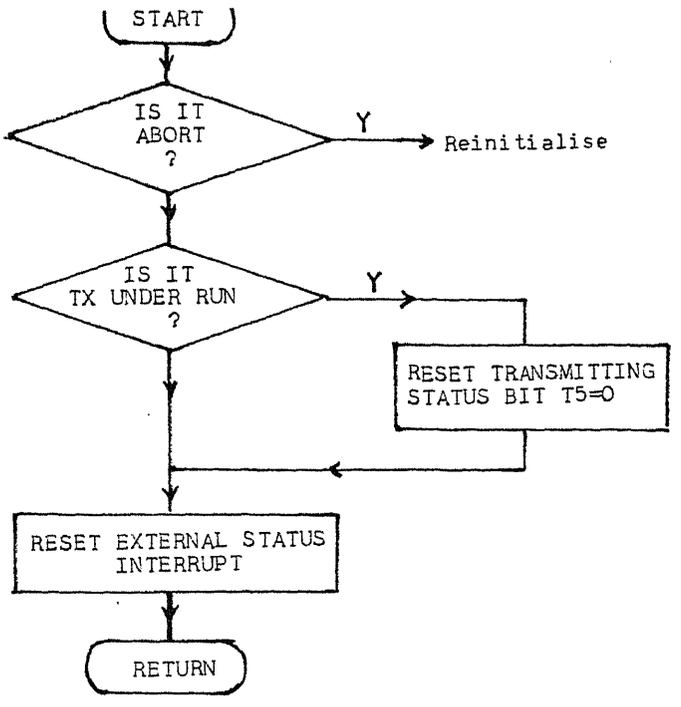


Fig.4(3) Flow chart-EXTERNAL STATUS ISR

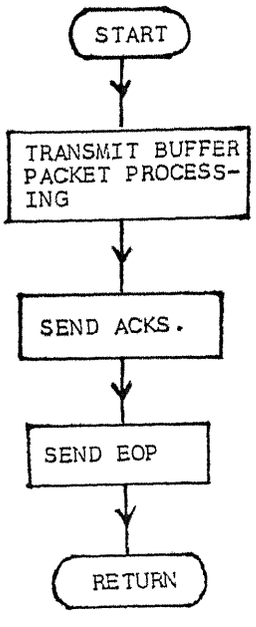


Fig.4(4) Flow chart. TRANSMIT LOOP.

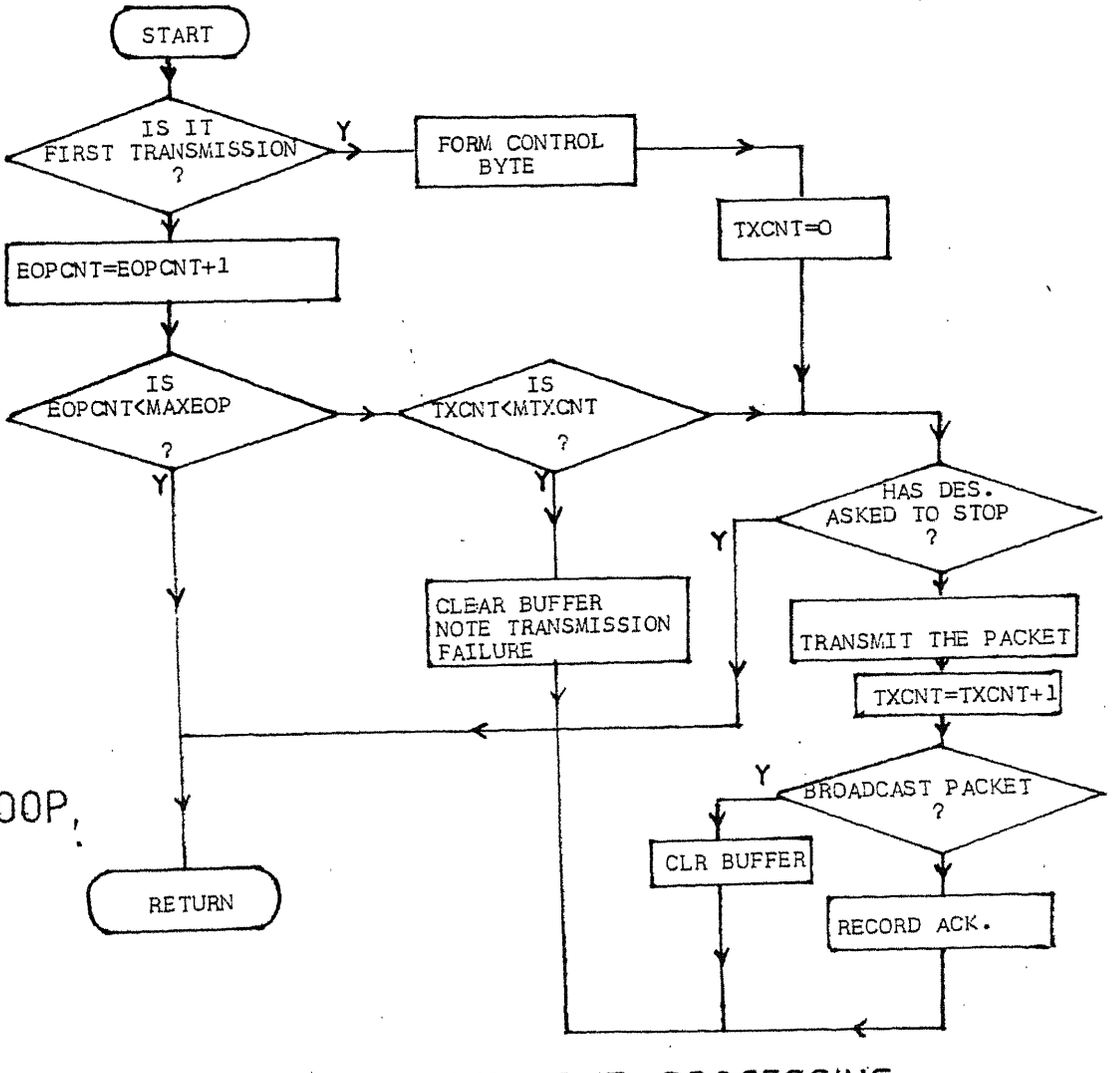


Fig.4(5) Flow chart-TRANSMIT BUF. PKT. PROCESSING

4.3.5 Transmit Loop

Transmit loop involves 3 different operations :

(i) transmitting one's own packet, if any
(ii) transmitting pending acknowledgements, if any
and (iii) forwarding of the EOP packet.

During transmission of one's own packet, if the packet is being transmitted for the first time, the complete control byte is formed. After transmitting a packet, the node waits for the acknowledgement for this packet until a fixed number of EOPs are received. If the acknowledgement does not arrive after this period, the node times out and again retransmits the packet. This process is repeated until a specified maximum number of transmissions are over. If the acknowledgement is not forthcoming even after this, transmission failure is assumed and the buffer cleared. Anytime the acknowledgement is received, the transmit buffer is cleared to mark successful transmission.

First location in the acknowledgement buffer gives the number of acknowledgements pending and subsequent locations the addresses of nodes to which they are pending. Acknowledgements are sent to all these nodes, in the acknowledgement routine.

Finally EOP is forwarded by the node. After EOP is forwarded the EOP received status bit (R7) is reset and the node

again goes to repeat mode until next EOP is received.

4.3.6 Processing of Circular Buffer

In this routine we do the processing of the packets stored in the circular buffer. This involves

(i) repetition of all packets which are not addressed to self
(ii) inputting of self addressed packets to receive buffer
(iii)repetition and inputting of broadcast packets
and (iv) noting the acknowledgements in the self addressed acknowledgement packets.

Packets addressed to self are accepted only if

(i) receive buffer and acknowledgement buffer are not full and (ii) the packets are not duplicates of previously received ones.

Otherwise the packets are simply rejected. When the packet is accepted or the packet is a duplicate one, the acknowledgement is recorded for latter transmission. Acknowledgements are sent for duplicate packets also,so as to take care of situations in which acknowledgement for the packet received get lost. These 'delayed' acknowledgements for the earlier received packet informs the source node that the packet has been correctly received.

Broadcast packets are input to receive buffer only if

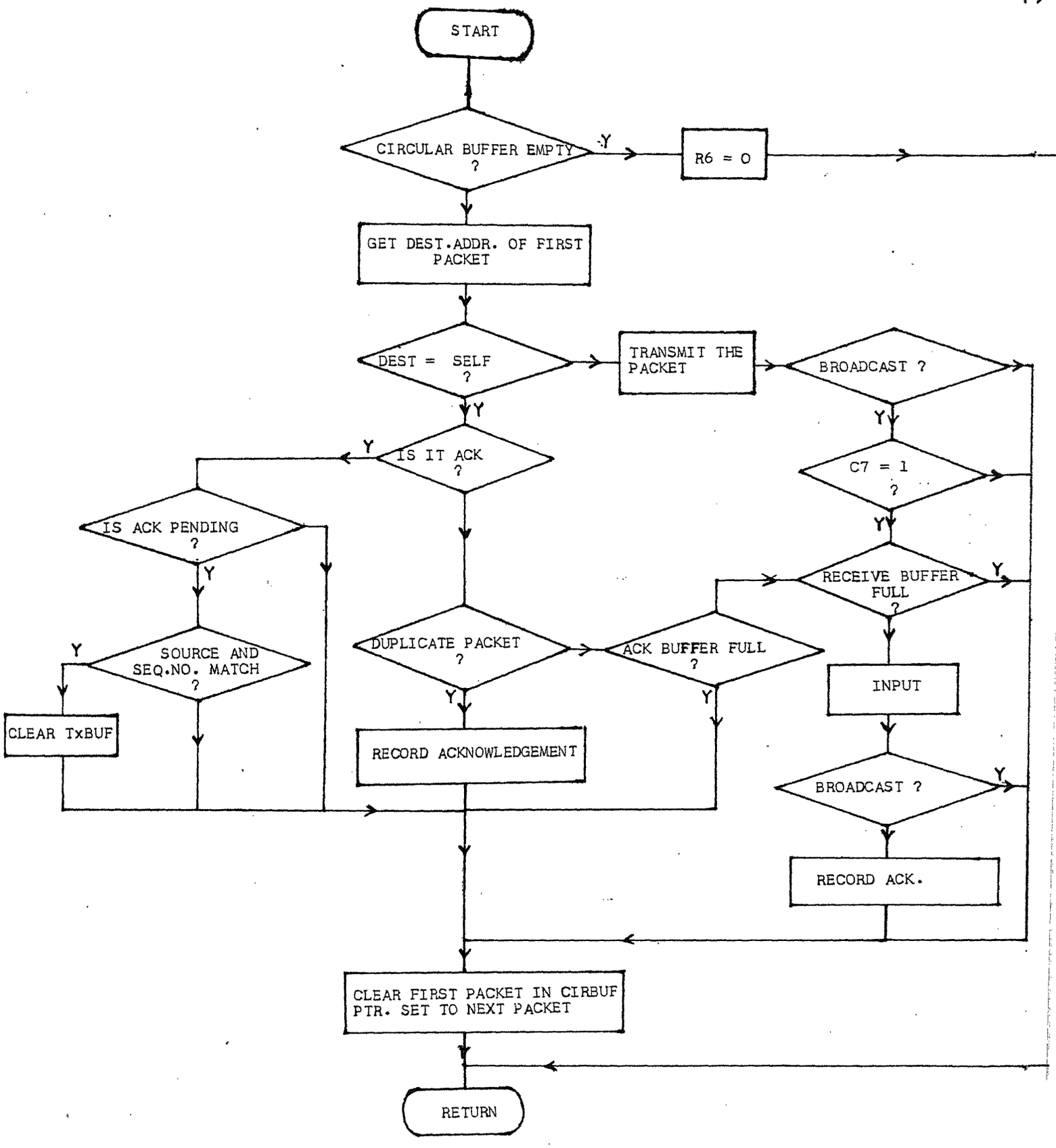


Fig.4(6) Flow chart-PROCESS CIRCULAR BUFFER

(i) receive buffer is not full
and (ii) the broadcast packet is in its second leg, i.e., retransmitted from primary.

Acknowledgements are accepted if (i) acknowledgement is pending and (ii) source and sequence number of the acknowledgement match . Otherwise the packet is simply rejected.

4.4 ERRORS AND REINITIALISATION

Some of the unlikely error conditions, during a smooth working of the networking software are

(i) abort detection
(ii) CRC error of the received packet
and (iii) illegal type of the packet in the receive buffer.

Any of these errors indicate some malfunctioning of the loop and one would like to reinitialise the entire loop. It is interesting to note that in the present set up, when one node undergoes initialisation, it forces the entire loop to reinitialise. When 8274 gets initialised, its TxD connected to down loop nodes RxD undergoes transitions. This results in an abort detection in the down loop node and it also gets initialised. This process moves around the entire loop and entire loop gets reinitialised. The delay in the initialisation before interrupts are enabled, ensures that this process stops after the entire loop is initialised. So as

in the beginning,the nodes start networking software after some delay,and packet transmission and reception restarts. So when any of the aforementioned errors are detected at some node, it simply goes back to initialisation and waits for the entire loop to reinitialise and then starts loop action.

CHAPTER 5

CONCLUSIONS

Two IMP nodes, connected by twisted pairs, were found to be successfully transmitting and receiving packets from each other. One of this was programmed as primary and the other as secondary. Though presently only two nodes were tried out, this set up can support upto 255 nodes, i.e., 1 primary and 254 secondaries. Intel's Intellec Series III-MDS and HP-2647A Graphics terminal were hooked onto the loop as Host stations through RS-232 interfaces.Communication between the host processors was also found to be successful. Each host could address its packets to a particular node or all nodes. Any self addressed packets or broadcast packets were successfully received by each node. Presently the loop is being operated at 192 KBs and IMP Host interface at 2.4 KBs.

As mentioned previously, possible upgradationsof the system include an EOP detection circuit and a DPLL type clock recovery. If channels 2 and 3 of 8237 DMA controller are used for reception and transmission, channels 0 and 1 can be used in memory to memory transfer mode. This requires minor modifications in the 8274-8237 interface and software. Memory to memory transfers by DMA can be used for inputting packets from

circular buffer to receive buffer and similar operations, thus ensuring faster response. [Note: 8237 channels 2 and 3 cannot be used for memory to memory transfers]. Also one would like to replace the IITK workstations by simple 8085 processors along with minimal amount of hardware that is necessary. Using 8085 AH-2 and 8237A-5 (minimum clock time = 200 ns) with 10 MHz crystal oscillator, will upgrade the loop data rates to 312.5 KBs. Only change in the hardware required for this will be to add another divide by 2 counter before the clock inputs of 8253 timer.

Presently, the IMP interconnections are through twisted pairs. Simple TTL inverters are used as line drivers and receivers. But in a practical environment, where transmission lengths are more, one will have to go in for RS-422 type drivers and receivers. Coaxial cables, with better noise immunity and physical ruggedness may be preferred over twisted pairs at higher data rates. Note that data rates upto 880 KBs are possible with appropriate modifications in the clock recovery circuit.

In applications like file transfer, one will have to transmit long strings of characters. Limitations on the available buffer size and the desire to have a fast response time lead to the need for segmentising the long strings of characters. The need to do error checking on a frame by frame basis, while seeing to it that the inevitable retransmissions

do not take too long, also points to packetisation. Each node splits up these long strings into a series of packets before transmission. Receiving node reassembles (depacketises) these series of packets to get the complete string. In situations like this, where both source and destination are fixed, it is desirable to avoid unnecessary packet overheads. This calls for call connect/terminate facility at each node. Call connect/terminate facility can be provided by suitably modifying the IMP software.

With these aforesaid upgradations, one hopes the present set up will be a satisfactory choice, to link PCs, workstations, terminals etc. with sufficiently high data rates and reasonable costs.

REFERENCES

(1) Ken Hardwick and William Federlick, 'Local Networking - the missing link emerges', Local Network Handbook, pp 11-19.

(2) Andrew, S. Tanenbaum, 'Computer Networks'.

(3) Paul E. Green, 'An introduction to network architectures and protocols', IEEE Trans. on Comm., April 1980, pp 412-423.

(4) David E. Carlson, 'Bit oriented data link control procedures', IEEE Trans. on Comm., April 1980, pp 455-467.

(5) John Beaston, 'LSI devices control loop mode SDLC data links', Local Network Handbook, pp 165-172.

(6) A. Hopper and D.J. Wheeler, 'Maintenance of ring communication systems', IEEE Trans. on Comm., April 1979, pp 759-760.

(7) C. Kenneth Miller and David M. Thompson, 'Making a case for token passing in local networks', Local Network Handbook, pp 48-53.

(8) Microprocessors and Peripherals Handbook, Intel, 1985.

APPENDIX A

PACKET STRUCTURE AND DATABASES

IMPLEMENTATION USING 8274-MPSC and 8237-DMA CONTROLLER (DMA BASED SDLC MODE OF OPERATION)

PACKET STRUCTURE :

FLAG/DEST/CNTL/SRCE/DISPOS/LEN1/LEN2/..INF../CRC/CRC/FLAG
Note that this format agrees with the standard SDLC format, where we have DEST and CONTROL bytes followed by INFO bytes and CRC bytes. Here the first 4 bytes of the INFO. FIELD are used as SOURCE, DISPOS., LEN1 and LEN2 respectively. FLAG (7E hex pattern) is used as the packet delimiter.

DATA TRANSPARENCY is ensured by the automatic zero bit insertion/deletion (ZBI) of 8274. ERROR checking and detection are done automatically by 8274, using CCITT-CRC code (i.e., modulo poly. $x^{16}+x^{12}+x^5+1$.

DISPOSITION BYTE is meant for carrying packet header for higher layer processing, wherever relevant e.g., D7, D6 = 00 indicate ASCII packet.

LEN1, LEN2 specify the no. of message bytes in a packet, LEN1 being the most significant byte and LEN2 the least. Total length of any packet is [LEN1, LEN2]+8 (hex) bytes. This may vary from 8 to MAXLEN. MAXLEN is a constant declared in the DEFINITION FILE. Currently this is equal to 263.

ADDRESSES for various NODES:-

PRIMARY (P)	00 (hex)
SECONDARIES (S)	01 - FE (hex)
BROADCAST PACKETS	FF (hex)

A maximum of 255 nodes (1 Primary and 254 Secondaries) may be connected in the LOOP. The address FF (hex) is used to denote Broadcast packets.

CONTROL BYTE FORMAT

C7 = 1 P -->,S, second leg of S-->P-->S, second leg of the broadcast packets from S.

= 0 S -->P, first leg of S-->P-->S,P-->P, first leg of the broadcast packets from S.

C6 = SEQ. No. of packet for NON-ACK PACKETS (i.e., if C5=1)
SEQ. No. of the packet being acked for ACK PACKETS (i.e. if C5=0)

C5 = 1 for NON-ACK PACKETS
= 0 for ACK.PACKETS
= SEQ.No. of the packet being acknowledged for PIGGY BACKED cases (i.e., if C1, C0=11).

C4,C3,C2 left unused for future expansions

C1,C0 TYPE of the packet
00 - DATA without piggy backing
01 - COMMAND packet
10 - STATUS packet
11 - PIGGY BACKED DATA packets

DATABASES

1. CONSTANTS TO BE SPECIFIED

SELF 1 byte constant specifying the addr of the NODE. This will be 00 (hex) for the PRIMARY and any no. from 01 to FE (hex) for SECONDARIES . (Each secondary will have a unique address.)

MAXLEN 2 byte constant specifying the MAXIMUM allowed LENGTH of a packet, including CRC bytes. Currently this is specified as 263.

MTXCNT constant specifying the MAXIMUM permitted No. of TRANSMISSIONS of a packet. If ACK. is not received even after this many transmissions, that is taken as a TRANSMISSION FAILURE and the buffer cleared.

MAXEOP constant specifying the no. of EOPs to be received before a packet is retransmitted if ACK. is not received.

2. VARIABLES, POINTERS, BUFFERS etc.

USER TABLE — RAM area from USTAB to USTAB+FF (hex) (i.e., 256 bytes) carrying information about the status of other NODES in the network. All locations are initialised to 80 (hex) at start up.

USER TABLE at NODE 'i' summarises for NODE 'i' its previous transactions with each of other NODES. For NODE 'j', the corresponding table entry will be at USTAB+j (hex). The entry at USTAB+i (hex), for NODE 'i' is used when self test by loop back is attempted (by PRIMARY or any of the SECONDARIES). Entry at USTAB+FF (hex) is also initialised to 80 (hex), but otherwise left untouched.

At NODE 'i', the entry at USTAB+j (hex) has the following interpretations.

USTAB7 — SEQ. NO. (0/1) of the next packet to be send to the NODE 'j', if no acks. are pending and SEQ. NO. of the packet for which the ack is pending, if ack. is pending.

USTAB6
USTAB5 — left unused

USTAB4=1 if the last transmission to NODE 'j' failed even after repetitions. (Resetting may be done by the higher layers.)
=0 otherwise

USTAB3 — SEQ.NO. (0/1) of the last correctly received packet from the NODE 'j' (The packet may or may not have been passed on to the higher layers.) Note that the BROADCAST packets do not have any SEQ. NO. and these are not under consideration, when we say last packet received correctly.

USTAB2
USTAB1 left unused

USTABO=1 if the NODE 'j' has currently requested to stop transmission
=0 otherwise

ROUTE 1 byte RAM location (R7-RO) reporting the status of operation at the NODE. ROUTE is initialised to OO (hex) at start up.

The interpretation of the various bits in ROUTE is as follows.

R7 =1 EOP packet has currently been received and not yet forwarded
=0 otherwise

R6 =1 CIRCULAR BUFFER currently not empty. (i.e. CIRBUF contains packet/packets to be processed by the NODE.)
=0 otherwise

R5 =1 packet currently being processed not to be acknowledged (This bit is being used in PROCESS$CIRBUF routine.)
=0 otherwise

R4 =1 currently inputting to RECEIVE BUFFER
=0 otherwise

R3 =1 received packet waiting in RCVBUF to be output to the USER process.
=0 otherwise

R2 =1 USER process currently inputting to TRANSMIT BUFFER
=0 otherwise

R1 left unused

RO =1 ACKs waiting in ACK. BUFFER for transmission.
=0 otherwise

TXSTAT 1 byte RAM location (T7-TO) indicating the status of TRANSMISSION at the NODE. TXSTAT is initialised to OO (hex) at start up.

The interpretation of the various bits in TXSTAT is as follows

T7,T6 left unused

T5 =1 NODE currently in TXMIT mode (The transmission may be that of a message packet generated by the NODE or ACK packet or a REPEAT packet or EOP packet.)
=0 otherwise

T4 =1 previous packet transmission failed (i.e. ACK was not received) even after repetitions.
=0 otherwise

T3 =1 designated TXBUF for inputting from the USER process currently full. (Do not input now.)
=0 otherwise. (USER process may input to TXBUF.)

T2 =1 TXBUF1 for transmit to the network and TXBUFO for USER process input.
=0 vice versa.

T1 =1 ACK. not yet received for the previous packet sent.
=0 otherwise (i.e. no pending ACKs)

TO =1 packet ready in proper TXBUF for being sent
=0 otherwise (i.e. no packet waiting to be sent in the TXBUF designated for transmission).

TXCHAR 1 byte RAM location carrying the first byte (i.e. DEST. byte) of a packet to be loaded by the CPU to the transmit buffer of 8274-Ch. A to initiate DMA action for transmission.

TPTOUT 2 byte RAM location carrying the starting address for DMA during a transmission. Before initiating DMA by loading the byte at TXCHAR to 8274-Ch.A, for a transmission, addr, reg. of 8237-Ch.1 is initialised with the contents of TPTOUT.

LENTT 2 byte RAM location carrying the no. of message bytes in the packet to be transmitted. The terminal count in 8237-Ch.1, for a transmission will be equal to (LENTT)+4 (hex).

TXBUF 2 buffers TXBUF0 and TXBUF1 each of length MAXLEN-2 (hex.). Starting addr. for these 2 buffers are also given by the 2 byte constants TXBUF0 and TXBUF1 respectively.

Organisation of TXBUF0 and TXBUF1

DESTINATION
CONTROL
SOURCE
DISPOSITION
LEN1
LEN2
:
MESSAGE BYTES
:

USER PROCESS inputs to one buffer (if free), while the contents (if any) of the other buffer are transmitted on the network. Contents of the TXBUF are discarded (i.e. buffer is cleared) after the ACK is received or otherwise after the packets are repeated the specified maximum no. of times (i.e., MTXCNT). In the case of BROADCAST packets, the buffer is cleared after transmitting once.

TPTIN 2 byte RAM location carrying the start addr. of TXBUF to which USER process may input.

TPTTMP 2 byte RAM location carrying the addr. of LEN1 byte of a packet during USER process input to TXBUF.

ACKND 1 byte RAM location containing the addr. of the NODE from which ACK. is expected.

TXCNT 1 byte RAM location, indicating the no. of times, the current packet in TXBUF designated for transmission, has been transmitted.

EOPCNT 1 byte RAM location, carrying the count of EOPs received after the last transmission of the current packet in the TXBUF.

ACKBUF 32+1 byte buffer in RAM containing the address of the NODES to which ACK. is pending. First location in the ACKBUF contains the no. of ACKs pending. This location is initialised to 00 (hex) at start up. Addresses of the NODES to which ACKs are pending are stored from second byte onwards in ACKBUF. Note that ACKBUF is also a 2 byte constant giving the start addr. of ACK$BUFFER.

Organisation of ACK$BUFFER

No. of ACKS
NODE ADDR. 1
NODE ADDR. 2
:

ACKCNT 1 byte RAM location carrying the no. of ACKs still left to be transmitted, while SEND$ACK routine is executed.

ACKPAK 5 byte buffer carrying the portion of the ACK. packet to be transferred by DMA controller, during transmission (i.e., second byte onwards). ACKPAK is also a 2 byte constant giving the start addr. of this buffer. Note that the byte at ACKPAK is the control byte and is modified according to SEQ. No. of the packet being acknowledged, prior to sending of each ACK packet.

ACKPAK FORMAT:- CNTL/SELF/08/00/00

CNTL byte = 40 if SEQ. No.=1
= 00 if SEQ. No.=0 for SECONDARIES and PRIMARY LOOP BACK
= 00 if SEQ. NO.=1
= 80 if SEQ. NO.=0 for PRIMARY (ord)

RCVBF 4 buffers of length MAXLEN-1 (hex) no. of bytes each viz. RCVBF0, RCVBF1, RCVBF2 and RCVBF3. Note that RCVBF0, RCVBF1, RCVBF2 and RCVBF3 are also 2 byte constants giving the starting addr. of the respective buffers. The first byte in each is the BUFSTAT byte, giving the status of the buffer. This is followed by the complete received packet (if any), except for the CRC bytes.

Organization of RCVBFs

BUFSTAT
DESTINATION
CONTROL
SOURCE
DISPOS
LEN1
LEN2

:

MESSAGE

:

BUFSTAT is organised as follows

B7-B3		left unused
B2	=1	if the buffer is being emptied
	=0	otherwise
B1	=1	if the buffer is being filled
	=0	otherwise
B0	=1	if the buffer is full (or being emptied)
	=0	if the buffer is empty (or being filled)

RCVINP 2 byte pointer in RAM indicating the start addr. of the RCVBF to which the next incoming packet is to be put.

RCVOTP 2 byte pointer in RAM indicating the start addr. of the RCVBF from which the next packet is output to the USER process.

BUFPNI — 1 byte pointer in RAM indicating the RCVBF to be used next to input from the CIRBUF. This byte is initialised to 00 (hex) at start up.

BUFPNO — 1 byte pointer in RAM indicating the RCVBF from which the next packet, if any, is to be output to the USER process. This byte is initialised to 00 (hex) at start up.

Organisation of BUFPNI (BI7-BI0)a and BUFPNO(BO7-BO0)

BI7-BI2 and BO7-BO2 unused

BI1-BI0 and BO1-BO0	= 00	if RCVBF0
	= 01	if RCVBF1
	= 10	if RCVBF2
	= 11	if RCVBF3

CIRBUF — CIRCULAR BUFFER to which incoming packets from the network are put for later processing has a length 2K bytes for primary and 1K bytes for secondaries. Correctly received packets, retained in the buffer are forwarded to the next NODE or input to the RCVBF or both, as appropriate, during processing. CIRBUF is also a 2 byte constant giving start addr of this buffer.

CRBFSA — 2 byte constant with the same value as CIRBUF

CRBREA — 2 byte constant giving the end addr. of CIRBUF. The value will be equal to CIRBUF+2047 for primary and CIRBUF+1023 for secondaries.

SACRBF — 16 byte buffer in RAM carrying the START ADDR. of the successive PACKETs retained in CIRBUF. SACRBF is also a 2 byte constant giving the start addr of this buffer.

Note that SACRBF is also essentially a 'circular' buffer. Each time a packet is received correctly in CIRBUF, the lower and higher bytes of the start addr. of the currently received packet will be stored in the locations addressed by SACRBF+2*CCRBF and SACRBF+2*CCRBF+1 (hex) respectively. Start addr. of the next packet to be processed in CIRBUF is at locations SACRBF+2*FCRBF and SACRBF+2*FCRBF+1 (hex).

CCRBF 1 byte pointer in RAM indicating the currently received packet in CIRBUF.CCRBF is initialised to OO (hex) at start up.

FCRBF 1 byte pointer in RAM indicating the first packet (i.e. next packet to be processed) in CIRBUF. FCRBF is initialised to OC (hex) at start up.

Organisation of CCRBF (CB7-CBO) and FCRBF(FB7-FBO)

CB7-CB3 and FB7-FB3	unused
CB2-CBO and FB2-FBO	a no. from O to 7 denoting the respective buffer.

Each time a new packet is accepted in CIRBUF, CCRBF is incremented by 1 (modulo 8). Each time processing of a packet in the CIRBUF is completed, FCRBF is incremented by 1 (modulo 8). Note that CIRBUF, at a time, can have a maximum of 8 packets (including the EOP packet).

CIRINP 2 byte RAM location carrying the start addr. for the next packet reception (i.e. next free location in CIRBUF).

CIROTP 2 byte RAM locatinn carrying the start addr. for the next outputting (to RCVBF) from CIRBUF.

CHOTC 2 byte RAM location carrying the allowed length (i.e., no. of bytes) of the next packet reception. This length will be equal to MAXLEN or no. of FREE LOCATIONS in CIRBUF, whichever is less. In no case this value will be less than the length of an EOP packet (i.e. 9).

DEST 1 byte RAM location carrying the destination address of the packet being input during USER$INPUT.

USTABI 1 byte RAM location carrying the USTAB byte for destination during USER$INPUT.

DESTP 1 byte RAM location carrying the destination address of the packet being processed during PROCESS$CIRBUF.

CNTLP 1 byte RAM location carrying the control byte during PROCESS$CIRBUF.

SRCEP 1 byte RAM location carrying the source address during PROCESS$CIRBUF

EOPPAK 6 byte buffer in RAM or ROM where the portion of the EOP packet to be loaded by DMA, during transmission of EOP (i.e., second byte onwards) is stored. EOPPAK is also the start addr of the buffer.

Organisation of EOPPAK buffer is as follows :

A1/ØØ/Ø8/ØØ/Ø1/7F (all values in hex)

APPENDIX B

NETWORKING ALGORITHM

VARIOUS ROUTINES USED BY THE SECONDARIES

MASTER$LOOP

(1) If the USER wants to input PACKETS (i.e., if the USER types ' 'N' then CALL USER$INPUT

(2) If R3=1, then CALL USER$OUTPUT

(3) CALL CHECK$STAT

(4) Go to (1)

CHECK$STAT

(1) If T3=1, T1=Ø, TØ=Ø and R2=Ø then continue else RETURN
(2) T3=Ø, T2 complemented, TØ=1
(3) TXCNT=Ø, EOPCNT=Ø
(4) RETURN

TXMIT$LOOP

(1) CALL TX$PACKET
(2) If RO=1 then CALL SEND$ACK
(3) CALL SEND$EOP
(4) RETURN

TX$PACKET

(1) If T1,TO = 00 then RETURN
= 01 then go to (2)
= 10 then go to (7)
(2) If T2=0 then DESTT=(TXBUF0)
else DESTT=(TXBUF1)
(3) USADRT=USTAB+DESTT
(4) If (USADRT)0 = 1 then RETURN
else continue
(5) CALL FORM$CONTROL

```
(6) Go to (11)
(7) EOPCNT = EOPCNT+1
(8) If EOPCNT < MAXEOP then RETURN
    else continue
(9) If TXCNT < MTXCNT then continue
    else go to (19)
(10) If (USADRT)0 = 1 then RETURN
    else continue
(11) If T2=0 then TXCHAR = (TXBUF0), TPTOUT = TXBUF0+1
    else TXCHAR = (TXBUF1), TPTOUT = TXBUF1+1
(12) LENTT = (TPTOUT+3), (TPTOUT+4)
(13) CALL TXMIT
(14) If TXCHAR=FF (hex), then continue
    else go to (17)
(15) T1=0, T0=0
(16) RETURN
(17) ACKND = TXCHAR, TXCNT = TXCNT + 1, EOPCNT = 0
(18) RETURN
(19) T4 = 1, T1 = T0 = 0, (USADRT)4 = 1
(20) RETURN
```

SEND$ACK

```
(1) LENTT=ØØØØ (hex) TPTOUT=ACKPAK
(2) ACKPNT=ACKBUF, ACKCNT=(ACKBUF)
(3) If ACKCNT=Ø, then go to (10)
    else continue
(4) ACKPNT=ACKPNT+1
(5) TXCHAR = (ACKPNT)
(6) If (USTAB+TXCHAR)3=1 then (TPTOUT)=4Ø (hex),
    else (TPTOUT) = ØØ (hex)
(7) CALL TXMIT
(8) ACKCNT = ACKCNT-1
(9) Go to (2)
(10) RØ = Ø, (ACKBUF) = Ø
(11) RETURN.
```

SEND$EOP

```
(1) TXCHAR = FF, TPTOUT = EOPPAK, LENTT = 0001 (hex)
(2) CALL TXMIT
(3) R7 = 0
(4) RETURN
```

FORMØCONTROL

(1) If T2=0 then TPTOUT = TXBUF0+1
    else TPTOUT = TXBUF1+1
(2) (TPTOUT)7 = 0, (TPTOUT)6 = (USADRT)7, (TPTOUT)5 = 1
(3) RETURN

TXMIT

(1) T5 = 1
(2) Reset transmit CRC generator
    Reset Tx INT/DMA pending
(3) Addr. reg. of CH1 of 8237 = TPTOUT
    Count reg. of CH1 of 8237 = LENTT + 4
(4) Unmask CH1 of 8237
(5) Load TXCHAR to 8274 CH. A DATA reg.
(6) Reset Tx Under run/EOM latch
(7) Wait until T5 = 0
(8) RETURN

PROCESSØCIRBUF

(1) If T5=Ø and R4=Ø continue
    else RETURN
(2) If FCRBF = CCRBF then continue
    else go to (5)
(3) R6 = Ø
(4) RETURN
(5) R5=1, CIROTP = (SACRBF+2*FCRBF)
(6) FCRBF=FCRBF+1
(7) DESTP=(CIROTP), CNTLP=(CIROTP+1), SRCEP=(CIROTP+2)
(8) If DESTP = SELF then go to (12)
    else continue
(9) TXCHAR = DESTP TPTOUT = CIROTP+1
    LENTT = (CIROTP+4), (CIROTP+5)
(10) CALL TXMIT
(11) If DESTP=FF (hex), (CNTLP)7=1
    and SRCEP≠SELF then go to (18)
    else RETURN
(12) If C5=Ø then continue
    else go to (15)
(13) If T1=1, ACKND=SRCER and
    (USTAB+ACKND)7=C6
    then T1=Ø, TØ=Ø, (USTAB+ACKND)7 complemented
(14) RETURN
(15) If (ACKBUF)=2Ø (hex) then RETURN
    else continue

```
(16) R5=Ø
(17) If C6<> U3 then continue
     else go to (23)
(18) If (BUFPNI)1,Ø=ØØ then RCVINP=RCVBFØ
                    =Ø1 then RCVINP=RCVBF1
                    =1Ø then RCVINP=RCVBF2
                    =11 then RCVINP=RCVBF3

(19) If BUFSTAT=(RCVINP)2,1,Ø=ØØØ
     then continue
     else RETURN
(20) CALL CIRBUF$TO$RCVBUF
(21) If (RCVINP)=SELF
     then (USTAB+SRCEP)3 complemented
(22) If R5=Ø then (ACKBUF)=(ACKBUF)+1,
     ACKPTT=ACKBUF+(ACKBUF),
    (ACKPTT)=SRCEP
(23) RØ=1
(24) RETURN
```

CIRBUFTORCVBUF

```
(1) R4=1
(2) (RCVINP)2,1,0=010, RCVINP=RCVINP+1
(3) DESTR=(CIROTP), LENTR=(CIROTP+4), (CIROTP+5)
(4) LENGTH=LENTR+6
(5) (RCVINP)=(CIROTP)
(6) RCVINP=RCVINP+1,CIROTP=CIROTP+1, LENGTH=LENGTH-1
(7) If LENGTH = 0 then continue
    else go to (5)
(8) R3=1, (RCVINP)2,1,0 = 011
(9) BUFPNI=BUFPNI+1
(10) R4=0
(11) RETURN
```

USER$INPUT

```
(1) If T3 = 1 then continue
    else go to (4)
(2) Signal USER that new packet cannot be accepted
(3) RETURN
(4) If T4=1 then continue
    else go to (8)
(5) Inform the USER the failure of previous transmission
    and query whether to continue or not
(6) If USER wants to continue then continue
    else RETURN
(7) T4 = 0
```

```
 (8) If T2=1 then TPTIN=TXBUF0
     else TPTIN=TXBUF1
 (9) R2=1
(10) Query  USER for DEST of the packet and store it at DEST
(11) (TPTIN) = DEST
(12) If DEST=FF (hex) then go to (21)
     else continue
(13) USTABI=(USTAB+DEST)
(14) If USTABIØ=1 then continue
     else go to (17)
(15) Signal USER that DEST requested to stop transmission
(16) go to (32)
(17) If USTABI4=1 then continue
     else go to (21)
(18) Signal USER that previous transmission to current DEST
     failed and query whether to continue or not
(19) If the USER wants to continue then continue
     else go to (32)
(20) (USTAB+DEST)4=Ø
(21) TPTIN=TPTIN+1, (TPTIN) = ØØ
(22) TPTIN=TPTIN+1, (TPTIN) = SELF
(23) TPTIN=TPTIN+1, (TPTIN) = DISPO = 08 (hex)
(24) TPTIN=TPTIN+1, (TPTTMP)= TPTIN,TPTIN=TPTIN+1
     LENIN=0000 (hex),
(25) Query USER for DATA
(26) If no more DATA (i.e., USER types <ESC>) then go to (30)
     else continue
(27) (TPTIN -)=DATA byte (from user),TPTIN=TPTIN+1 LENIN=LENIN+1
(28) If LENIN<MAXLEN then go to (25)
     else continue
(29) Signal and stop USER from providing more DATA
(30) T3=1
(31) (TPTTMP)=LEN1=MSB$LENIN, (TPTTMP+1)=LEN2=LSB$LENIN
(32) R2=0
(33) RETURN
```

USER$OUTPUT

```
 (1) If Host asserts CTS to IMP then continue
     else RETURN
 (2) If BUFPNO=ØØ then RCVOTP=RCVBFØ
                =Ø1 then RCVOTP=RCVBF1
                =Ø1 then RCVOTP=RCVBF2
                =11 then RCVOTP=RCVBF3
 (3) RXPT=RCVOTP
 (4) If (RCVOTP)Ø =0 then continue
     else go to (7)
```

```
(5) R3=Ø
(6) RETURN
(7) (RCVOTP)2,1,Ø=101
(8) RCVOTP=RCVOTP+1,
    DEST=(RCVOTP)
(9) If DEST=FF (hex)
    then inform USER that it is a broadcast packet
(10) RCVOTP=RCVOTP+1
(11) If (RCVOTP)1,Ø=ØØ then continue
    else ERROR*go to reinitialise*
(12) RCVOTP=RCVOTP+1, SOURCE=(RCVOTP)
(13) Output SOURCE to the USER
(14) RCVOTP=RCVOTP+1
(15) If (RCVOTP)7,6=ØØ continue
    else ERROR*go to reinitialise*
(16) RCVOTP=RCVOTP+1, NUM$MSB=(RCVOTP)
(17) RCVOTP=RCVOTP+1, NUM$LSB=(RCVOTP)
(18) NUM=NUM$MSB,NUM$LSB (2 bytes)
(19) Output NUM to the USER
(20) If NUM=Ø then go to (24)
    else continue
(21) Output (RCVOTP) to USER
(22) NUM=NUM-1, RCVOTP=RCVOTP+1
(23) Go to (20)
(24) Signal end of packet to the USER
(25) (RXPT)2,1,Ø,=ØØØ
(26) BUFPNO=BUFPNO+1
(27) Go to (1)
```

INTERRUPT$SERVICE

```
(1) Save all registers
(2) Read the contents of CHANNEL B READ REG.2 of 8274
    VECTOR = contents of CH.B RD.RG2 of 8274
(3) If VECTOR = 1C (hex) then go to SPRCVD$ISR
              = 14 (hex) then go to EXTSTS$ISR
(4) End of interrupt
(5) Enable interrupts
(6) Restore all registers and RETURN
```

SPRCVD$ISR

```
(1) Read the contents of CHANNEL A READ REG.1 of 8274
(2  SFLAG = contents of CH.A RD RG1 of 8274
(2) If S5 = 1 (i.e. RECEIVE OVERRUN) then go to (27)
    else continue
(3) If S7 = 1 (i.e. EOF RECEIVED) then go to (6)
    else continue
(4) End of interrupt (CHA)
(5) Go to (37)
```

(6) If S6=1 (i.e. CRC ERROR) then ERROR*go to reinitialise
else continue
(7) If (CIRINP)=FF (hex) and CIRINP+1)=A1 (hex) and
(CIRINP+2)=00 (hex) and (CIRINP+6)=7F (hex)
then continue
else go to (10)
(8) R7 = 1
(9) Go to (27)
(10) If FCRBF=CCRBF+1 (modulo 8) then to go (27)
else continue
(11) NXTLOC=CIRINP+LENT+6 (next free location)
FREEND=CRBFEA-NXTLOC (no of free loc at end)
(12) If FREEND<MAXLEN then go to (15)
else continue
(13) CHOTC=MAXLEN, CIRINP=NXTLOC
(14) Go to (26)
(15) CBSTRT=(SACRBF+2*FCRBF)(start addr. of first pkt)
(16) FREBEG=CBSTRT-CIRBUF (No. of free loc. at beg.)
(17) If FREBEG>FREEND then continue
else go to (24)
(18) If FREBEG>or = MAXLEN then continue
else go to (21)
(19) CIRINP=CBSTRT, CHOTC=MAXLEN
(20) Go to (26)
(21) If FREBEG<9 (i.e. length of EOP packet)
then go to (27)
else continue
(22) CIRINP=CBSTRT, CHOTC=FREBEG
(23) Go to (26)
(24) If FREEND<9 then go to (27)
else continue
(25) CIRINP=NXTLOC, CHOTC=FREEND
(26) CCRBF=CCRBF+1, R6=1,
(SACRBF+2*CCRBF)=CIRINP
(27) Error reset
End of interrupt
(28) Enable interrupts
(29) CALL PROCESS$CIRBUF
(30) If R7=1 continue
else go to (33)
(31) If R6=1 go to (29)
else continue
(32) CALL TXMIT LOOP
(33) Enable interrupt on next receive character
Reset receive CRC checker

(34) Addr. reg. of CHØ of 8237=CIRINP
Count.reg.of CHØ of 8237=CHØTC-1
(35) Unmask channel 0 of 8237 for next reception
(36) Restore all registers and RETURN
(37) Enable interrupts
(38) Restore all registers and RETURN

EXTSTSØISR

(1) Read the contents of the CHANNEL A READ REG. 0 of 8274
EFLAG=contents of CH. A RD.RG.1 of 8274
(2) If E7=1 (i.e. ABORT detected) then ERROR go to reinitialise
else continue
(3) If E6=1 (i.e. TXMIT UNDER RUN) then continue
else go to (5)
(4) T5 = 0
(5) Reset external status interrupt
End of interrupt
(6) Enable interrupts
(7) Restore all registers and RETURN

INITIALISATION

(1) Initialise all relevant pointers, variables etc.
ROUTE = 0, TXSTAT = 0, TXCNT =0, EOPCNT= 0,
BUFPNI=0, BUFPNO = 0, FCRBF =0, CCRBF = 0
RCVBF0=0, RCVBF1 = 0, RCVBF2=0, RCVBF3= 0
ACKBUF=0, CIRINP = CIRBUF, CHOTC = MAXLEN,
(SACRBF), (SACRBF+1) = CIRBUF
USTAB to USTAB + FF (hex) = 80 (hex)

Partial packets for ACKs and EOP i.e. ACKPAK and
EOPPAK respectively are stored.

(2) Write appropriate JMP INSTRUCTIONS at locations to
which CPU branches when interrupt occurs
A 580C write JMP INTERRUPTØSERVICE
(Note: At 002C we already have JMP 580C)
(3) Initialise 8274 in SDLC mode, CH. A DMA and CH. B.INTR,
NON VECTORED STATUS AFFECT VECTOR mode of interrupts,
interrupt on FIRST RECEIVE CHAR etc.
(4) Initialise 8237 - CHO and CH1 in SINGLE TRANSFER mode,
CHO for reception and CH1 for transmission, all
registers initialised, CONTROLLER enabled and CHO
unmasked CHO addr. reg = CIRBUF, count reg = MAXLEN-1
(5) Wait until the all NODES in the LAN are initialised to
this point (a fixed time delay)

(6) Reset any pending interrupts of 8274
(Read the contents of CH. B RD. RG .2, Error reset,
Reset external status interrupt, End of interrupt

(7) Display the WELCOMING MESSAGE
(8) Unmask and enable RST 5.5 intr.
(9) Go to MASTER$LOOP

ROUTINES WHICH ARE DIFFERENT IN THE CASE OF PRIMARY

MASTER$LOOP

(1) If R7=1 continue
    else go to (6)
(2) If R6=1 continue
    else go to (5)
(3) CALL PROCESS$CIRBUF
(4) Go to (2)
(5) CALL TXMIT$LOOP
(6) If USER pressed '^N' then CALL USER$INPUT
(7) If R3=1, then CALL USER$OUTPUT
(8) CALL CHECK$STAT
(9) Go to (1)

SEND$ACK

(1) LENTT=ØØØØ (hex) TPTOUT=ACKPAK
(2) ACKPNT=ACKBUF, ACKCNT=(ACKBUF)
(3) If ACKCNT=Ø then go to (13)
    else continue
(4) ACKPNT=ACKPNT+1
(5) TXCHAR=(ACKPNT)
(6) If TXCHAR=SELF then go to (9)
    else continue
(7) If (USTAB+TXCHAR)3 = 1
                then (TPTOUT)=CØ (hex)
                else (TPTOUT)=ØØ (hex)
(8) Go to (10)
(9) If (USTAB)3=1 then (TPTOUT)=4Ø (hex)
    else (TPTOUT)=ØØ (hex)
(10) CALL TXMIT
(11) ACKCNT=ACKCNT-1
(12) Go to (2)
(13) RØ=Ø (ACKBUF)=Ø
(14) RETURN

FORM$CONTROL

(1) If T2=Ø then TPTOUT=TXBUF0+1
    else TPTOUT=TXBUF1+1
(2) If DEST. of the packet = SELF continue
    else go to (5)
(3) (TPTOUT)7=Ø
(4) Go to (6)
(5) (TPTOUT)7=1
(6) (TPTOUT)6=(USADRT)7
    (TPTOUT)5=1
(7) RETURN

SPRCVD$ISR

(1) Read the contents of CHANNEL A
READ REG.1 of 8274
SFLAG=contents of Ch.A RD.RG1 of 8274
(2) If S5=1 (i.e., RECEIVE OVERRUN) then go to (29)
else continue
(3) If S7=1 (i.e. EOF RECEIVED) then to go (6)
else continue
(4) End of interrupt (CHA)
(5) Go to (32)
(6) If S6=1 (i.e. CRC ERROR) then ERROR*go to reinitialise*
else continue
(7) If (CIRINP)=FF (hex) and (CIRINP+1)=A1 (hex) and
(CIRINP+2)=ØØ (hex) and (CIRINP+6)=7F (hex)
then continue
else go to (10)
(8) R7=1
(9) Go to (29)
(10) If (CIRINP+1)7=1 go to (29)
else continue
(11) (CIRINP+1)7=1
(12) If FCRBF=CCRBF+1 (modulo 8) then go to (29)
else continue
(13) NXTLOC=CIRINP+LENT+6 (next free location)
FREEND=CRBFEA-NXTLOC (No. of free location at the end)
(14) If FREEND<MAXLEN then go to (17)
else continue
(15) CHØTC=MAXLEN, CIRINP=NXTLOC
(16) Go to (28)
(17) CBSTRT=(SACRBF+2*FCRBF) (start addr. of first pkt.)
(18) FREBEG=CBSTRT-CIRBUF (No.of free loc. at beg.)
(19) If FREBEG>FREEND then continue
else go to (26)
(20) If FREBEG≥MAXLEN then continue
else go to (23)
(21) CIRINP=CBSTRT, CHØTC=MAXLEN
(22) Go to (28)
(23) If FREBEG<9 (i.e., length of EOP packet)
then go to (29) else continue
(24) CIRINP=CBSTRT, CHØTC=FREBEG
(25) Go to (28)
(26) If FREEND<9 then go to (29)
else continue
(27) CIRINP=NXTLOC, CHØTC=FREEND
(28) CCRBF=CCRBF+1, R6=1
(SACRBF+2*CCRBF)=CIRIND

(29) Error reset
End of interrupt
Enable intr. on next receive char.
Reset receive CRC checker
(30) Addr. reg. of CHØ of 8237=CIRINP
Count.reg. of CHØ of 8237=CHØTC-1
(31) Unmask CHANNEL Ø of 8237 for next reception
(32) Enable interrupts
(33) Restore all registers and RETURN.

INITIALISATION

(1) Initialise all relevant pointers, variables etc.
ROUTE = 0, TXSTAT = 0, TXCNT = 0, EOPCNT = 0
BUFPNI = 0, BUFPNO = 0, FCRBF = 0, CCRBF = 0
RCVBF0 = 0, RCVBF1 = 0, RCVBF2= 0, RCVBF3 = 0
ACKBUF = 0, CIRINP = CIRBUF, CHOTC = MAXLEN,
(SACRBF), (SACRBF+1) = CIRBUF
USTAB to USTAB + FF (hex) = 80 (hex)

Partial packets for ACKs and EOP i.e. ACKPAK and EOPPAK respectively are stored.
(2) Write appropriate JMP INSTRUCTIONS at locations to which CPU branches when interrupt occurs
At 580C write JMP INTERRUPTØSERVICE
(Note: At 002C we already have JMP 580C)
(3) Initialise 8274 in SDLC mode, CH. A DMA and CH.B INTR, NON VECTORED STATUS AFFECT VECTOR modeof interrupts, interrupt on FIRST RECEIVE CHAR etc.
(4) Initialise 8237 - CHO and CH1 in SINGLE TRANSFER mode, CHO for reception and CH1 for transmission,all registers initialised, CONTROLLER enabled and CHO unmasked
CHO addr. reg = CIRBUF, count reg = MAXLEN - 1
(5) Wait until the all NODES in the LAN are initialised and have already started networking software
(6) Reset any pending interrupts of 8274
(Read the contents of CH B RD. RG.2, Error reset, Reset external status interrupt, End of interrupt)
(7) Display the WELCOMING MESSAGE
(8) Unmask and enable RST5.5 intr.
(9) SENDØEOP
(10) Go to MASTERØLOOP